KARAKTERFELISMERÉS

Vizsgadolgozat

Készítette:
Faragó Csaba
III. éves programozó matematikus szakos hallgató

Témavezető:
Csirik János
tanszékvezető egyetemi tanár

JATE, Informatikai Tanszékcsoport
Szeged, 1999. május 11.

Tartalomjegyzék

Ábrák jegyzéke

1. Bevezetés

1.1. Kézírásfelismerés

A kézírásfelismerés területének hosszú ideig kis jelentőséget tulajdonítottak a mintafelismerés területén belül. Csak néhány általános célú konferencián volt lehetőség arra, hogy a kutatók publikálják az eredményeiket.

A számítógépek gyors fejlődésének következtében időközben felmerült az igény a gyors és kényelmes ember-számítógép kommunikációra. Az elmúlt időszakban komoly erőfeszítések voltak az új input eszközök kifejlesztése területén. Ennek a fejlesztésnek a legfontosabb eredményei a grafikus tábla, a szkenner, az egér, a fényceruza stb. Ennek ellenére azonban még ma sem tekinthetjük megoldottnak a problémát.

Ellentétben a múlt gyakorlatával, ma már igen sok ipari vállalat érdeklődését keltette fel a kézírásfelismerés területe. Ez az előrelépés nagyrészt annak köszönhető, hogy általános az igény a ceruza alapú ember-számítógép kapcsolatra.

A kézírásfelismerés területét három alterületre lehet felosztani: karakter- és számjegyfelismerés, írott szófelismerés és aláírásellenőrzés. Mindhárom esetben szükség van adatszerzésre, amely a későbbi feldolgozástól lényegében független problémát jelent.

Ez a dolgozat csak a karakterfelismerés területével foglalkozik.

1.2. Adatszerzés

Az adatszerző eszközöket két csoportra oszthatjuk: on-line és off-line eszközök. Az, hogy milyen eszközzel történik az adat bevitele, jelentős mértékben befolyásolja az alkalmazható módszereket. Egészen más technikákat lehet alkalmazni on-line eszközzel digitalizált szöveg esetén, mint off-line eszköz esetén.

Az off-line adatszerző eszközök közé a különböző szkennerek tartoznak, míg az on-line eszközök közül a digitalizáló táblák a legismertebbek.

Dolgozatomban off-line technikát alkalmaztam. Az szöveg digitalizálása lapszkenner segítségével történt.

Az adatszerzés területéhez szorosan kapcsolódik az adatok előfeldolgozása. Ez alatt a szöveg megkeresését, a betűk illetve szavak szegmentálását stb. értjük. Nagyon sok cikket jelentettek már meg és sok módszert dolgoztak ki ezzel kapcsolatban. E probléma nehézségi szintje hasonló, mint a felismerésé.

1.3. Karakter- és számjegyfelismerés

A karakter- és számjegyfelismerés a kézírásfelismerés legtöbbet kutatott részterülete. Ennek két fő oka van. Egyrészt a probláma hasonló a kézzel írott szöveg és a nyomtatott szöveg esetben is, és köztudott, hogy úgy az ipar, mint a tudomány érdekelt a karakterolvasás megoldásában. A másik oka pedig az, hogy a problémát klasszikus mintafelismerési technikák segítségével is meg lehet oldani.

Ennek a területnek a jelentősége fokozatosan nől, mivel a számítógépek elterjedésével egyre nagyobb az igény nagymennyiségű adatok gyors és pontos bevitelére.

A karakterfelismerési módszereket két nagy csoportba lehet osztani: mintaillesztési és struktúraelemző módszerek.

1.4. A diszkrét képmodell

A szakdolgozatom írásakor a diszkrét bináris képmodellt vettem alapul. Ez egy olyan modell, ahol a képpontok egy négyzetrács csúcsaiban helyezkednek el, és kétfajta értéket vehetnek fel: üres vagy teli.

Ezenkívül a „fekete" pontokra 8-összeköttetést vettem alapul, a „fehér" pontokra pedig a 4-összeköttetést. Ez azt jelenti, hogy két különböző fekete képpontra akkor mondjuk, hogy szomszédosak, ha a köztük levő távolság legfeljebb $\sqrt{2}$. Két különböző fehér pont akkor szomszédos, ha a köztük levő távolság pontosan 1.

Kétdimenzióban ez a legelterjedtebb modell. Erre dolgozták ki a legtöbb algoritmust.

1.5. Általában a programról

A szakdolgozatomhoz kapcsolódó program egy `BMP` fájlt dolgoz fel. Kézzel, nyomtatott nagybetűkkel írott betűket próbál felismerni. A program egyelőre csak az angol ábécé nagybetűire koncentrálódik, magyar ékezetes betűket még nem tud kezelni. A programban be kell állítani a betűmagasságot és -szélességet, a függőleges és vízszintes eltolást, valamint a dőlésszöget is.

A program eddigi legjobb felismerési aránya 95%-os volt. Ez egy Pentium személyi számítógépen kb. 10 perc alatt lefutott. A minta 30 sorból állt, ennyiszer voltak leírva az angol ábécé nagybetűi.

A programot `Linux` operációs rendszer alatt írtam, `C++` nyelven. A program lefordításához a `GNU C++` fordítót használtam. Csak olyan könyvtárakat használtam, amelyek a legtöbb platformon rendelkezésre állnak (`stdio.h`,

`stdlib.h`). Tehát egy szabványos `C++` fordítóval a legtöbb platformon le tudjuk fordítani a programot.

2. Előfeldolgozás

2.1. Digitalizálás

A karakterfelismerő programom jelenlegi verziója nem képes arra, hogy a papíron megkeresse a szöveget. Tehát manuálisan kell eljutni addig a pontig, ahol már külön vannak választva az egyes betűk. Az adatgyűjtés egy egyszerű módjához folyamodtam: egy francia kockás lapra leírtam hússzor az angol ábécé nagybetűit. Ezt `300 dpi`, `TrueColor` minőségben beszkenneltem, majd a kapott képet 8 biten kódolt, sűrítés nélküli `BMP` formátumra hoztam. Tehát más képformátumokkal, illetve a `BMP` más lehetőségeivel nem foglalkoztam. Ezen hiányosságokat a későbbiekben be szeretném pótolni.

A program először a `BMP` fájlból készít egy `raw` nevű fájt, amely a `BMP` fejléc nélküli bináris tartalma. Fekete pontnak számít az, amely `RGB` összetevője (ezeknek összege) egy bizonyos korlát alatt marad. Kék toll használata esetén csak a piros és zöld összetevőket kell figyelembe venni. A programkészítés ezen szakaszában sem törekedtem „ideális” program elkészítésére: a `raw` fájlban a fehér pixelek helyén 0 áll (a 48-as `ascii` kódú karakter), míg a fekete pixelek helyén 1-es szerepel (`ascii` 49). Ezt azért csináltam így, mert ily módon könnyebben tudom ellenőrizni a program futását. A program (egy) végeleges formába öntésekor természetesen ezt érdemesebb bináris módon megoldani.

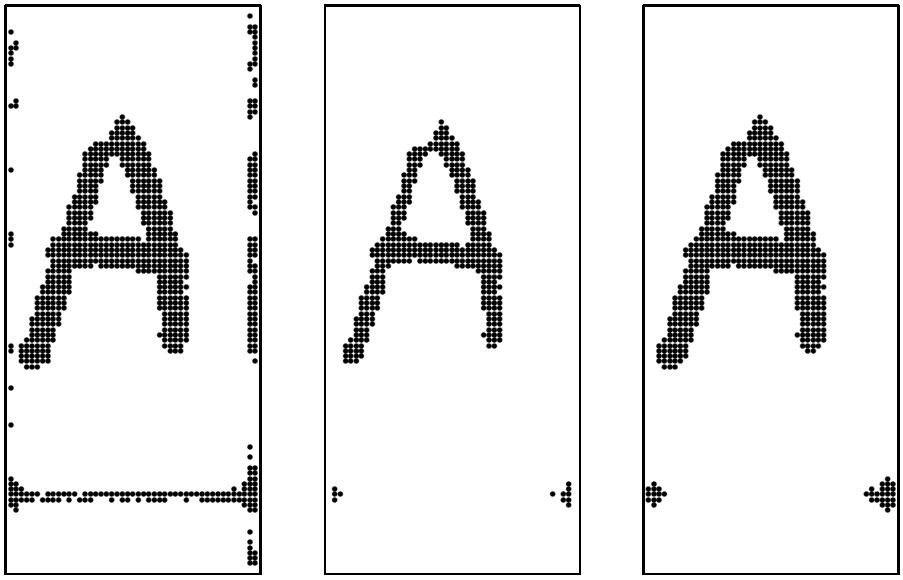
A következő lépés a `raw`-ból a betűk kinyerése. Itt van a programom legsúlyosabb hiányossága: nincsen automatizálva a betűk kikeresése. Tehát a betűszélességet és -magasságot, a vízszintes és függőleges eltolást, a sorok és oszlopok számát, valamint az esetleges ferdeséget manuálisan kell megadni. Ez egy olyan nagyvolumenű részprobléma, amivel most nem foglalkoztam. A jövőben ezt is korrigálni szeretném. A program eredménye *oszlopokszáma* $*$ *sorokszáma* darab új állomány, amelyben ott vannak a betűk bináris formában (48, 49 `ascii`). Ez a példánkban $20 * 26 = 520$ állományt jelent. Az állományok nevei: `(oszlopsorszám: 2 karakter)x(maga a betű: 1 karakter)`. A későbbiekben ezt sem kell kimenteni lemezre, hanem inkább a memóriában érdemes tárolni.

2.2. A `BMP` fájl felépítése

A `BMP` fájl eszközfüggetlen formában tárolja a képinformációkat. A fejléc részben a képméretre és a kódolás típusára vonatkozó információk vannak. Egy pixel 1, 4, 8 vagy 24 biten tárolódik. Amennyiben nem 24 biten tárolódik, a fejléc információkat a színtábla követi. 8 biten kódolt esetben 256 szín piros, zöld és kék összetevői vannak megadva. 24 bites esetben nincs szükség színtáblára.

Ezenkívül lehetőség van még `LZV`-tömörített formátumban is tárolni az adatokat.

A programomban csak a 8 biten kódolt, nem tömörített `BMP` formátummal foglakoztam. Ezért az input képet először mindig ilyen formára kell hozni.



1. ábra. Az eredeti objektum, erózió, dilatáció

2.3. Szűrés

Ezen lépés inputja a `raw` fájlból kinyert „nyers” karakterek, outputja pedig ennek morfológiailag megszűrt formája. A morfológiai szűrés megmagyarázásához be kell vezetni két fogalmat: az erózió és a dilatáció fogalmát.

`Erózió`: általános esetben ez azt jelenti, hogy a fekete pontok által alkotott objektumból elveszünk valamennyit, „körberágjuk”, mégpedig azon fekete képpontokat „ítéljük halálra”, amelyeknek egy bizonyos sugarú környezetében fehér pont található. Ez most azt jelenti, hogy azon pontokat töröljük a képről, amelyeknek a 8-szomszédos környezetében van fehér pont. (Egy pont 8 szomszédos környezetén a közvetlenül mellette és az átlósan mellette levő pontokat értjük. Tehát ez megfelel a sakkirály lépéslehetőségeivel). Az erózió általában nem őrzi meg a topológiát.

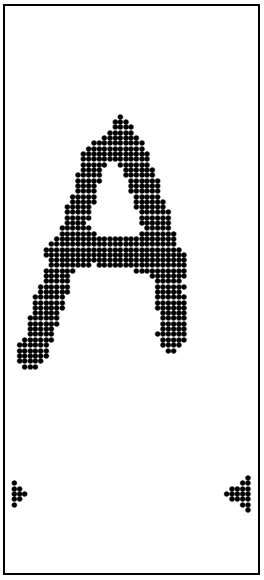
`Dilatáció`: ez az erózióval szimmentrikus művelet. Tehát most az objektum „hízlalásáról” van szó. Azokat a fehér pontokat fogom feketére váltani, amelyeknek a 8-szomszédos környezetében van legalább egy fekete pont. Jól látható, hogy ez a művelet sem őrzi meg a topológiát.

Mind az eróziót mind a dilatációt párhuzamosan kell végezni.

A `morfológiai szűrés` egy erózió, két dilatáció, és végül ismét egy erózió végrehajtása. A szűrés tehát szintén nem őrzi meg a topológiát.

Jó szűrési módszer az egy dilatáció, két erózió, egy dilatáció is. A szűrés lényege az, hogy eltüntesse a képről azokat a kicsi fekete foltokat, amelyek az előfeldolgozás során keletkeztek, eltüntesse azokat a kicsi fehér üregeket, amelyek a betűk belsejében találhatók, és szintén nem játszanak szerepet, valamint megszüntesse a „gyanús", túl vékony összeéréseket.

A szűrés az eredményét egy fájlba írja. A fáljnév vége `s`.



2. ábra. A betű szűrt képe

Vigyázni kell a szűrés alkalmazásával, mivel szétszaggathatunk olyan vékony betűket is, amelyeket egyben kell tartani.

Ráadásul ezt a lépést nem szükséges mindig végrahajtani, előfordul, hogy már elég „szépek" a betűk. A program további hiányossága az, hogy a szűrés szükségességének az eldöntése sincsen automatizálva.

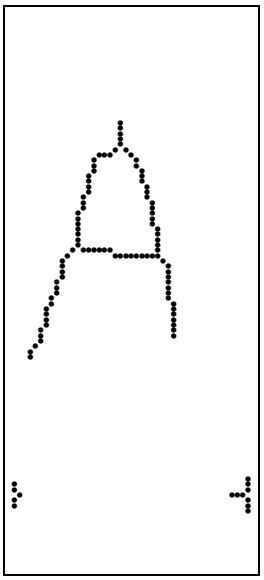
2.4. Vékonyítás

A vékonyítás inputja a szűrt kép, outputja pedig egy olyan kép, ahol a vonalak 1 pixel vastagságúak. A vékonyítás már megőrzi a topológiát, ezenkívül a végpontokra is ügyel. Egy vékonyító algoritmussal szembeni követelmény ezen kívül az, hogy az eredmény lehetőleg az eredeti objektum közepén helyezkedje el, mintegy kijelölve annak vázát.

Én egy kétfázisú párhuzamos algoritmust valósítottam meg. Az iterációk egyes lépéseiben azokat a pontokat kell törölni, amelyek eltávolítása nem sérti a topológiát, viszont ügyelni kell arra, hogy a törlendő pont le legyen végpont. Ezen feltételek lokálisan eldönthető, mindössze a vizsgált képpont 8 szomszédját kell figyelembe venni. A két fázis addig kell felváltva végrehajtani, amíg van mit törölni.

A program forrásában megfigyelhető egy tecnikai trükk: egy 256 méretű tömbben jelöltem meg azokat az eseteket, amelyekben törölni kell. Egy képpontot körülvevő 8 pont ugyanis 256 féle alakot vehet fel. Ezen pontok állapontai bináris számként is felfoghatjuk. Tehát elég a táblázatban megnézni azt, hogy az éppen vizsgált pontot az aktuális fázisban törölni kell-e vagy sem. Az algoritmus végén még szükség van egy kis korrekcióra, mert néhány olyan pixel is benne marad a képben, amelyet törölni kellene.

Az eredményt fájlba írom, aminek a neve `v`-re végződik.

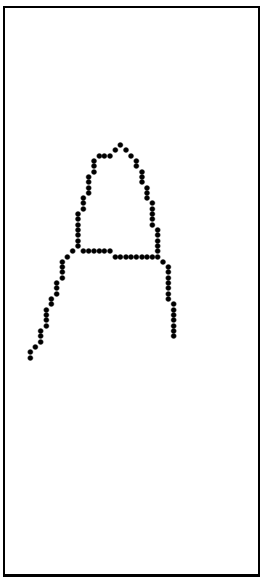


3. ábra. A betű vékonyított képe

2.5. A vékonyított kép utófeldolgozása

Ebben a fázisban a kép olyan részeit törlöm, amely zavaró lehet a felismerés során. Először törlöm a „keretet” (a kockás papír miatt maradhat „szemét” a betű szélein) – ezt a lépést már előbb is érdemes lehet végrehajtani – még nincs benne az aktuális programban.

Itt törlöm ki azokat az apró „kinövéseket”, amelyek a vékonyítás után a képen maradtak, viszont általában semmilyen szerepet sem játszanak, zavaró az ottlétük. A legfeljebb 5 pixelnyi „kinövéseket” törlöm. Ezt technikailag úgy oldom meg, hogy 5 lépésen keresztül törlöm azokat a pontokat, amelyeknek legfeljebb egy szomszédjuk van, majd ismét 5 lépésen keresztül a megmaradt végpontokból visszaiterálok. Tehát így a lényeges részek továbbra is eredeti hosszukban maradnek meg. Ugyanaz a technika megfigyelhető itt is, mint a vékonyításnál.



4. ábra. A betű lényeges részei

Ezenkívül megkeresem a kép közepén levő objektumot, azt körbejárom, és ahova nem tudok eljutni, azt törlöm. Ezáltal összesen egy darab összefüggő objektum marad a képen. Ez hibát okozhat akkor, ha az eredeti képen a betű nem összefüggő. (Ez gyakran előfordul pl. a `T` vagy `E` betűk esetén.)

Az eredményt olyan állományokba írja, amelyeknek a neve `1`-re végződik.

3. Felismerés

Ha semmit sem rontottuk el, akkor most már a betű vékonyított, felesleges és zavaró részektől mentes állapotban áll rendelkezésünkre. Elérkeztünk a leglényegesebb részhez: a felismeréshez. Ezt a következőképpen valósítottam meg: a betű körbejárásával kialakítok egy vektort. A mintabetűk szintén vektor formájában állnak rendelkezésemre. Az éppen vizsgált vektort mindegyik mintabetű vektorával összehasonlítom, és amelyikre legjobban hasonlít, arra mondom rá, hogy ez a megoldás. Ezt tehát két fő részre lehet osztani: a vektor meghatározása és az összehasonlítások.

3.1. A vektor kialakítása

Az adott betű vektorát annak vékonyított képe körbejárásával kapjuk. Ez azt jelenti, hogy elindulunk egy pontból, megjegyezzük, hogy ott már jártunk, majd megvizsgáljuk, hogy milyen irányba tudunk továbbhaladni, és feljegyezzük azt, hogy merre léptünk. Ezt addig ismételjük, még be nem járunk minden pontot. A körbejárás technikai megvalósítása jóval egyszerűbb, mint az első ránézésre tűnik: a kezdőpontot betesszük egy verembe, majd amíg a verem nem üres, végrehatjuk a következő lépéseket: kiveszünk egy elemet a veremből, az összes még be nem járt szomszédját betesszük a verembe, miközben feljegyezzük a vektorban azt, hogy az éppen vizsgált ponttól milyen irányban található az éppen most a verembe rakott pont. Amennyiben egy olyan betűt járunk be, amely több különálló részből áll, úgy ezt az algoritmust addig kell ismételni, míg van be nem járt pont. Tehát miután lefutott, meg kell vizsgálni az egész képet, és ha találunk nem érintett pontot, akkor azt be kell tenni a verembe, és újra kell kezdeni az egészet.

Lényeges, hogy melyik pontból indulunk ki, amikor elkészítjük a vektort, valamint a haladás iránya is fontos. A programírás kezdeti szakaszában a legtöbbet a vektor helyes kialakításával foglalkoztam. Az algoritmus legelső futása során 63%-os pontosságot ért el. Ebben a verzióban a legfelső, azon belül pedig a legbaloldalibb képpontból indultam ki. Egy adott pont szomszédait fentről lefelé, azon belül pedig balról jobbra vizsgáltam. Az ezt követő programverziókban fokozatosan javítottam ezt, amelyről a későbbiekben az 4.1 fejezetben még szó lesz.

A program a jelenlegi verziójában már több vektort is kialakítok egy betűből. Minden egyes olyan pontból körbejárom a betűt, amelynek nem pontosan két szomszédja van. Tehát a végpontokból és a csomópontokból kiindulva végzem el a műveletet.

Az alábbi néhány vektor az ábrán látható `A` betű körbejárásainak az eredménye.

```
0 6 6 6 6 7 5 5 5 6 5 6 6 5 6 6 5 6 6 6 6 5 6 6 6 6 7 6 8 5 5 6 6 6 6 6 5
6 6 6 6 6 6 8 8 8 8 8 8 8 1 8 8 8 8 8 1 7 2 2 2 2 2 2 3 2 2 3 2 2 2 3 2 2 3
4 4 3 7 7 6 6 6 7 6 6 7 6 7 6 6 6 7 6 6 7 7 6 0 6 6 7 8 5 6 6 8 0 6 5 7 6 -1

0 2 2 2 2 2 2 1 2 2 2 2 2 2 1 1 8 2 7 2 2 2 2 1 2 2 2 2 1 2 2 1 2 2 1 2 1 1
1 7 2 2 2 2 7 8 8 7 6 6 7 6 6 6 7 6 6 7 6 6 6 6 6 6 7 5 4 4 4 4 4 5 4 4 4 4
4 4 4 7 7 6 6 6 7 6 6 7 6 7 6 6 6 7 6 6 7 7 6 0 6 6 7 8 5 6 6 8 0 6 5 7 6 -1

0 2 3 3 2 2 3 2 2 2 3 2 3 2 2 3 2 2 2 3 3 3 5 2 2 2 2 2 2 3 2 2 3 2 2 2 3 2
2 3 4 4 3 3 5 2 2 2 2 5 5 6 5 6 6 5 6 6 5 6 6 6 6 5 6 6 6 6 7 6 8 5 5 6 6 6
6 6 6 5 6 6 6 6 6 6 8 8 8 8 8 8 8 1 8 8 8 8 8 0 6 6 7 8 5 6 6 8 0 6 5 7 6 -1

0 6 5 7 6 0 6 6 6 6 7 5 5 5 6 5 6 6 5 6 6 5 6 6 6 6 5 6 6 6 6 7 6 8 5 5 6 6
6 6 6 6 5 6 6 6 6 6 6 8 8 8 8 8 8 8 1 8 8 8 8 8 1 7 2 2 2 2 2 2 3 2 2 3 2 2
2 3 2 2 3 4 4 3 7 7 6 6 6 7 6 6 7 6 7 6 6 6 7 6 6 7 7 6 0 6 6 7 8 5 6 6 8 -1

0 4 4 5 3 2 2 6 6 0 6 6 6 6 7 5 5 5 6 5 6 6 5 6 6 5 6 6 6 6 5 6 6 6 6 7 6 8
5 5 6 6 6 6 6 6 5 6 6 6 6 6 6 8 8 8 8 8 8 8 1 8 8 8 8 8 1 7 2 2 2 2 2 2 3 2
2 3 2 2 2 3 2 2 3 4 4 3 7 7 6 6 6 7 6 6 7 6 7 6 6 6 7 6 6 7 7 6 0 6 5 7 6 -1

0 2 3 1 2 0 6 6 6 6 7 5 5 5 6 5 6 6 5 6 6 5 6 6 6 6 5 6 6 6 6 7 6 8 5 5 6 6
6 6 6 6 5 6 6 6 6 6 6 8 8 8 8 8 8 8 1 8 8 8 8 8 1 7 2 2 2 2 2 2 3 2 2 3 2 2
2 3 2 2 3 4 4 3 7 7 6 6 6 7 6 6 7 6 7 6 6 6 7 6 6 7 7 6 0 6 6 7 8 5 6 6 8 -1
```

3.2. A leghosszabb közös részsorozat

Két vektor leghosszabb közös részsorozatát a programban sokszor ki kell számolni. Egy betű esetén, azt feltételezve, hogy a mintából és a vizsgált betűből is két vektort veszek, több mint száz összehasonlítást kell elvégeznem. Egy vektor hossza az általam vizsgált példákban gyakran meghaladja a százat. Nagyon fontos tehát az, hogy a program ezen része a lehető leggyorsabb legyen.

Az általam megvalósított algoritmus aszimptotikus futási ideje $O(m * n)$, ahol m és n a két vektor hossza. Az algoritmus jellegét tekintve mohó. Az elve a következő: tekintsünk egy $m * n$-es mátrixot. E mátrix (i, j) értéke jelentse az első vektor első i és a második vektor első j eleme által alkotott vektorok leghosszabb közös részsorozatát. Minket az (m, n) pontban felvett érték érdekel, ez fogja adni a két teljes vektor leghosszabb közös részsorozatának a hosszát.

Tetszőleges (i, j) pontban ki tudjuk számolni az eredményt, ha ismerjük az $(i-1, j-1)$, az $(i, j-1)$ és az $(i-1, j)$ pontokban felvett értéket. Ebben az esetben az eredményt a következő három szám maximuma adja: Az $(i-1, j)$ helyen felvett érték, az $(i, j-1)$ helyen felvett érték, valamint az $(i-1, j-1)$ helyen felvett érték $+1$, ha az első vektor i. és a második vektor j. eleme megegyezik. Az első sor és az első oszlop esetén az előző sorra ill. oszlopra vonatkozólag nullát kell feltételeznünk.

Az algoritmus pszeudokódja:

Be: `vektor1[`m`]`, `vektor2[`n`]`

`matrix`: $m * n$-es, egészekből álló mátrix, amelynek van nulladik oszlopa és nulladik sora is.

```
1. matrix[0][0-n] = matrix[0-m][0] = 0
2. for i=1 to m
3.   for j=1 to n
4.     matrix[i][j] = max(matrix[i-1][j],matrix[i][j-1],
                          matrix[i-1][j-1]+
                          (vektor[i]==vektor[j])?1:0)
5. return matrix[m][n]
```

3.3. A betű felépítése alapján történő felismerése

A mintában szereplő betűkkel való összehasonlítások előtt érdemes megpróbálni felismerni a betűt a felépítése alapján. Ezt a `megprobalfelismerni` függvénnyel valósítottam meg. Tapasztalataim szerint ennek alkalmazása az eredményességet jelentősen nem befolyásolja, a futási időt viszont csökkenti, ugyanis egy ilyen vizsgálatnak a futási ideje elenyésző az összehasonlítások futási idejéhez képest.

A következőkben vázolom ennek a függvénynek a működését.

A függvény inputja a betű vékonyított és lényegtelen részektől lecsupaszított képe. A betűt körbejárjuk, miközben megszámoljuk a végpontokat (ezek azok a fekete képpontok, amelyek szomszédainak a száma pontosan egy) és a csomópontokat (ezek azok a fekete pontok, amelyek szomszédainak a száma több mint kettő). Miközben ezt végezzük, a betűt különálló egységekre osztjuk. Egy ilyen egységen belül már nem szerepelhet sem végpont, sem csomópont. Ezzel az algoritmussal sajnos bejönnek „parazita" partíciók is, amelyek semmilyen szerepet sem játszanak. Ezt oly módon kezeltem le, hogy a túl rövid (öt pixelnél rövidebb) részekkel nem foglalkoztam.

Ezek után az így kinyert részeket megpróbálom osztályozni. Ezt a `milyen` nevű függvény végzi. Az általam alkalmazott módszer középpontjában ismét a szakasz bejárása áll. A program összeszámolja, hogy a bejárás során hány alkalommal kellett lépni az egyes irányokba (nyolc ilyen lehetséges irány van: felfelé, jobbra-fel, jobbra, jobbra-le, le, balra-le, balra és balra-fel). Ezenkívül a szakaszt öt egyenlő részre osztja, és minden egyes ötöd rész esetén meghatározza az említett paramétereket. Az osztályozás az így összegyűjtött információk alapján történik.

A következő osztályokat különböztetem meg:

- függőleges egyenes szakasz: |
- vízszintes egyenes szakasz: —

- átlós egyenes szakasz (mindkét irányban): /, \
- metszet (összesen négy irányban elforgatva): $\cap$, $\cup$, $\subset$, $\supset$
- `L` mind a négy lehetséges elforgatása,
- `J` négy lehetséges elforgatása,
- `C` négy elforgatása,
- `S` négy lehetséges elforgatása,
- `V` négy lehetséges elforgatása: $<$, $>$, $\wedge$, $\vee$
- valamint egy olyan primitív, amely egy majdnem teljes kört zár be.

Ez így összesen 29 lehetséges osztály. A programban nincs mindegyik lehetőség implementálva, mivel mindegyikre nincs szükség. Ezenkívül elképzelhető, hogy a jövőben más alakú primitíveket is be fogok vezetni. A függvény ezenkívül külön osztályba sorolja a túl rövid darabokat. Abban az esetben, ha egy primitívet nem tudott osztályba sorolni, a visszatérési érték -1.

Az egyes primitívek vizsgálatakor ügyelni kell arra is, hogy általában nem tudjuk, melyik végpontból indultunk el. Tehát minden esetben két vizsgálatot kell tenni: mindkét végpont szerinti bejárást figyelembe kell venni.

Mielőtt rátérnék konkrét példák bemutatására, a forrás könnyebb megértéséhez egy kis technikai jellegű magyarázatot adok: az `otod` nevű tömbben tároltam az egyes ötöd részekben kiszámított, különböző irányú lépéseket. Ezen tömb dimenziói 6 és 9. Az irányokat a következőképpen jelöltem:

jobbra-fel 0	fel 1	balra-fel 2
balra 3		jobbra 5
balra-le 6	le 7	jobbra le 8

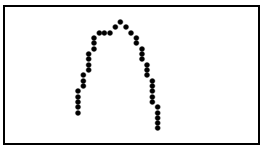
A könnyebb megvalósítás miatt hagytam ki a 4-et, amiben az adott ötöd rész hosszát tároltam. Az utolsó sorban a teljes szakaszra vonatkozó statisztikákat vettem fel. Így például az említett tömb második sorának hetedik eleme (`otod[1][6]`) azt mondja meg, hogy az adott irányú bejárás során a primitív második ötödében összesen hányszor kellett balra-le lépni, az `otod[4][4]` az utolsó ötöd rész hosszát adja meg, az `otod[5][1]` pedig azt mondja meg, hogy a teljes bejárás során hányszor léptünk felfelé.

Most néhány példát említek az általam szabott kritériumokról. Először megvizsgálom, hogy a primitív két végpontja y irányban elég távol van-e egymástól. Ha túl közel vannak, akkor már most kizárhatjuk azt az esetet, hogy függőleges szakaszról van szó. A kritérium nagyjából az, hogy a két végpont y koordinátája abszolút értékének a különbsége nagyobb legyen,

5. ábra. Példa vízszintes primitívre

mint a teljes primitív hosszának 90%-a (egy-két finomítástól eltekintve erről van szó). Ezen a ponton legfeljebb négy eset állhat fenn: a primitív lehet függőleges, átlós mindkét irányban, vagy nem tudunk dönteni. Azt, hogy ferde vagy függőleges osztályba soroljuk-e, a két végpont x irányú különbsége alapján döntjük el. Én a következő kritériumot használtam: ha az x irányú különbség legfeljebb az objektum hosszának a fele, akkor már csak függőlegesről lehet szó, ellenkező esetben pedig már csak átlósról. Ez azt jelenti, hogy körülbelül 65 és 115 fok közé esik a függőleges. Ahhoz viszont, hogy véglegesen rámondjunk bármit, még egy-két feltételnek teljesülnie kell. Függőleges esetben például meg kell akadályozni azt, hogy nagyon „cikkcakkos" legyen; ez az eddigi feltétlerendszerbe beleférne, viszont nem szabad elfogadni. Erre két korlátozást vezettem be: egyrészt maximalizáltam az egyik irányú átlós lépések számát (erős korlátot adtam az egyik irányra), valamint minimalizáltam a függőleges lépések számát (az összes lépés legalább kétharmadának függőlegesnek kell lennie). Érdemes ilyen szigorú feltételeket szabni, mivel még így viszonylag kevés olyan primitív van, amit ez a feltételrendszer visszautasít, és érdemes lenne függőlegesként kezelni, viszont ha olyan primitíveket is függőlegesként fogadna el, amely nem az, sokkal súlyosabb következményekkel járna, ugyanis könnyen helytelen eredményt kapnánk a felismerés során. Az átlós esetekre szintén megszorításokat vezettem be, bár azok nem annyira szigorúak, mint a függőleges primitív vizsgálatakor.

Vízszintes esetben hasonló kritériumokat szabtam.

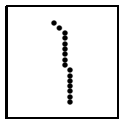


6. ábra. Példa $\cap$ primitívre

A következő lehetséges primitív a $\cup$. Ennek a feltételrendszerét úgy szemléltetem, hogy feltételezem, a bejárás balról jobbra történik. Az alapgondolat az, hogy az elején lefelé kell halani, utána jobbra, majd felfelé. Az eleje és a vége két-két ötödrészt jelent. Először a lefelé történő lépéseket vettem legnagyobb súllyal, a balra-le történő lépéseket kisebbel, és a jobbra-le történő lépéseket még kisebbel. Ezeken súlyozott értékek összegét minimalizáltam a

primitív összhosszúságának függvényében. A harmadik ötöd részben csak azt vizsgáltam meg, hogy jobbra többször léptem-e mint balra. Hasonló kritériumokat szabtam a fenti alak három lehetséges elforgatottjára is.

A következő vizsgált alakcsoport az `L` alak elforgatottjai. Ebben az esetben csak az első két és az utolsó két ötödrész játszik szerepet. A későbbiekben érdemes volna megtalálni a „természetes” szegmentációt, és így sokkal szigorúbb feltételeket lehetne adni ugyanolyan eredményességgel. Most csak azt teszem fel, hogy ez a pont valahol a primitív közepe tájékán van, ideális esetben a harmadik ötöd részben. Az egyenesség vizsgálatakor viszont szigorúbb feltételeket szabtam: megkövetelem, hogy legalább az egyik szára legyen „elég egyenes”. Ezt úgy valósítottam meg, hogy adott esetben még kisebb súlyokkal sem veszem fel az átlós irányokat, a feltétel tehát csak az adott egyenes irányába tett lépések számára vonatkozik. A másik szárra ekkor már enyhébb feltétel van érvényben; itt már kisebb súlyokkal ugyan, de felvettem a megfelelő átlós lépések számát is.



7. ábra. Példa függőleges primitívre

Ha még most sem jártunk eredménnyel, akkor azt vizsgáljuk meg, hogy a primitív hasonlít-e a `J` alakra, vagy annak tükörképére. Ezután következik a `C` és elforgatottjainak a vizsgálata, majd az `S` és végül a „majdnem kör” vizsgálata. Ezeknak a kritériumait már nem részletezem, a forrásból könnyen meg lehet állapítani.

4. A programról

4.1. A felismrőalgoritmus kialakulásának első lépései

A program első futó verziója 63%-os pontosságot ért el, az 520 megvizsgált betűből 332 betűt talált el. Ebben sem a kiindulópont meghatározásánál, sem a betűk körbenjárásánál, sem a felismerésnél nem használtam semmilyen heurisztikát. Ennek a verziónak a részletes statisztikái:

A 50%	B 55%	C 80%	D 85%	E 20%	F 35%	G 90%
H 85%	I 100%	J 100%	K 5%	L 95%	M 45%	N 75%
O 85%	P 70%	Q 35%	R 35%	S 100%	T 60%	U 40%
	V 30%	W 65%	X 50%	Y 75%	Z 95%	

Különösen rossz eredményt ért el a K betű esetén (a minta kivételével semmit sem ismert fel!), az E, az F, a Q, az R, az M, az U, a V, az A és a B betű esetén. Különös problémát jelent az, hogy nem tudjuk a hibákat általánosítani, ugyan arra a betűre többfajta hibát követ el. Jó példa erre a K betű: a tévedések között szerepelnek az I, R, V, L, Y, M és X betűk.

A következő programverziókban a bejárás irányát és a kezdőpontot változtattam meg. Először a bejárás irányára helyeztem a hangsúlyt, aminek az eredménye 70%, majd 72% lett. Az bejárási irány megváltoztatása következtében jelentős javulás következett be az A és B betűk esetén, 55% ill. 50%-ról 100% ill. 95%-ra nőtt az arány. Viszont csökkenést is tapasztaltam, és a K betűn ez a fejlesztés szintén nem segített.

A következőkben a kezdőpont kiválasztására koncentráltam. Ugyanis több betű alacsony felismerési arányának az oka az volt, hogy a mintában a kiindulási pont nem felelt meg az éppen vizsgált karakter indulási pontjával. Különösen igaz ez a K betűre: a mintában a jobb felső szárának a végpontja volt a bejárás kiindulási pontja, míg a többi esetben a bal függőleges rész felső pontja. A következőkben arra törekedtem, hogy a kiindulási pont lehetőleg egy olyan végpont legyen, amely a bal felső sarokhoz közel van. Ezáltal megoldódott a K betű súlyos problémája: a felismerési aránya a gyakorlatilag 0-ról 75%-ra nőtt. Jelentős javulás következett be az U arányát illetően is, ugyanis 40%-ról 95%-ra nőtt a pontos felismerés aránya. Az összesített arány is folyamatosan nőtt: először 77%, majd 80%, utána 83%, majd ismét egy kis módosítás következtében a vizsgált mintán 89%-os lett a pontos felismerés aránya.

Ekkor érkeztem el először arra a pontra, amikor a tévedések döntő többségének az okát már könnyen meg lehet érteni, túlnyomórészt az egymásra hasonlító betűket keverte össze a program.

Ezután átalakítottam a programot úgy, hogy egy betű vékonyított képéről

nem egy, hanem több vektort alakítok ki. A bejárást az összes csomópontból és végpontból elindítom, és mindegyik vektort mindegyik, a mintabetűben levő vektorral összehasonlítom. Ezáltal kiküszöbölhető az, hogy a nem megfelelő pontból történő indulás miatt rossz eredményt kapjunk. Viszont ez jelentősen lassítja a programot, ezért az összehasonlításra kerülő vektorok számát is korlátoztam. Tapasztalataim szerint két-két vektor összehasonlítása elegendő, az felett már nem javul jelentősen a pontosan felismert betűk aránya.

Sajnos az általam vizsgált mintán ennek a jelentősége nem mutatkozott meg, mivel a felismerési arány továbbra is kb. 90%-os maradt. Viszont más mintán lefuttatva a program előző verzióját csak kb. 80%-os arányt kaptam, az új módszerrel viszont ott is kb. 90%-os eredményt ért el.

4.2. A szemantikus felismerés bevezetése

Az összehasonlító algoritmust önmagában már nemigen lehet javítani. A hatékonyságot már csak elő- illetve utófeldolgozással lehet növelni. Éppen ezért bevezettem egy olyan függvényt, amely a betű vékonyított képét kielemezve próbálja felismerni a karaktert. Amennyiben ez sikerrel jár, már nincs szükség további összehasonlító vizsgálatokra. A program ezen részét már részletesen bemutattam az előző fejezetekben. Jelentős felismrési arányjavulás nem következett be, viszont szerintem ez a programom egyik legígéretesebb része.

Itt a fejlesztés során először csak kevés paramétert vettem figyelembe (például csak a végpontok és csomópontok száma), később viszont definiáltam az egyes primitíveket, amelyek jelenleg a szemantikus felismerési rész alapját képezik.

4.3. A program eredményessége

A legjobb eredményt idáig Müller Helga 30 soros mintáján érte el a program: 95%-os pontos felismerési arányt ért el. A részletes statisztikák az egyes betűkre a következők:

A 100%	B 100%	C 96%	D 73%	E 96%	F 100%	G 70%
H 100%	I 100%	J 96%	K 100%	L 100%	M 93%	N 96%
O 96%	P 93%	Q 96%	R 83%	S 100%	T 96%	U 93%
	V 100%	W 90%	X 100%	Y 100%	Z 96%	

Mindössze két dolgot szeretnék kiemelni: a G viszonylag alacsony arányát, aminek az oka az, hogy már a papíron is hibásan szerepel, a másik pedig a Q betű feltűnően magas, 96%-os aránya, ami azért meglepő, mert más minták

```
ABCDEFCHIJKLMNOPOPSTUVWXYZ
ABCDEFSHIFKLMNOPQRSTUVWXYZ
ABCPEFGHIJKLMNOPQRSTUVWXYF
ABCDEFSHIJKLMNOPQRSTUVWXYZ
ABCDEFGHIJKLMNOPQPSTOVWXYZ
ABCDEFGHIJKLMNOPQRSTUVWXYZ
ABCDEFGHIJKLMNOPQRSTUVVXYZ
ABCDEFSHIJKLMNOPQRSTUVWXYZ
ABCDEFSHIJKLMNOPQPSTUVWXYZ
ABCDEFGHIJKLMNOPQRSTUVWXYZ
ABCPEFGHIJKLMNOPQRSTUVWXYZ
ABCDEFGHIJKLMNOPQRSTUVWXYZ
ABCDEFSHIJKLMNOPQRSTUVWXYZ
ABCDEFGHIJKLMNPPQRSTUVWXYZ
ABCDEFGHIJKLMNOPQRSTUVWXYZ
ABCDEFGHIJKLMNOPQRSTUVWXYZ
ABCDFFSHIJKLMNOPQRSTUVWXYZ
ABCOEFGHIJKLMNOPQRSTUVWXYZ
ABCPEFGHIJKLMNOPQRSTUVWXYZ
ABZDEFGHIJKLMNODQRSTUVWXYZ
ABCPEFSHIJKLMIOPQRSTUVWXYZ
ABCDEFGHIJKLINOPQRSTUVWXYZ
ABCDEFGHIJKLMNOPQRSIUVWXYZ
ABCDEFGHIJKLMNOPQRSTUVWXYZ
ABCOEFGHIJKLMNORQPSTUVVXYZ
ABCOEFGHIJKLMNOPQRSTUVWXYZ
ABCOEFGHIJKLMNOPQRSTUVVXYZ
ABCDEFSHIJKLMNOPQRSTVVWXYZ
ABCDEFGHIJKLINOPQPSTUVWXYZ
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

8. ábra. A felismerés eredménye

esetén gyakran az `O` betűvel összetéveszti a program, és ezáltal csak körülbelül $50 - 60\%$-os arányt ér el.

Az ábrán látható eredmény mintája a hatodik sor volt.

4.4. A továbbfejlesztés lehetőségei

A programban számos ponton nyitva áll a lehetőség a továbbfejlesztésre. Ezek közül vannak olyanok, amelyek teljesen új algoritmust és heurisztikákat igényelnek, de olyanok is vannak, amelyek csupán a már kidolgozott heurisztikák továbbfejlesztését jelentik.

A program legnagyobb hiányossága az, hogy nem képes megtalálni a papíron levő szöveget. Ez egy igen nehéz feladat. Magába foglalja a szöveges

rész elkülönítését a többi elemtől, valamint az egyes karakterek szegmentációját. A szegmentáció alapja lehet az, hogy a szöveg téglalaprácsos papírra íródott, és az is könnyítést jelenthet, hogy ezáltal a betűk mérete nem ingadozik. Ezen tényezők figyelembe vételével sem triviális a probléma.

A program jelenleg csak a `BMP` bizonyos lehetőségeit ismeri. Hasznos lenne ennek a kibővítése. Ez viszonyalag nem nehéz, viszont rendkívül munkaigényes feladat.

A legígéretesebb továbbfejlesztési lehetőség a szintaktikus felismerés. A programot érdemes olyan szintre hozni, hogy ez a rész minél több betűre adjon pozitív visszatérő értéket, mivel egyrészt ez gyorsítja a program futását, másrészt a felismrés pontossága is növekedhet ezáltal, mivel a szintaktikus karakterfelismerés hatékonysága jóval meghaladja a statisztikus karakterfelismerés hatékonyságát. A programomban viszont nem veheti át teljes egészében a betű alakjának elemzése által történő felismerés az összehasonlításokkal történő módszer szerepét, mivel ez előbbi csak a „szép" betűkre tud pozitív választ adni, míg ez utóbbi viszont bármilyen „csúnya" betűre mond valamit.

Az egyes primitívek vizsgálata során nyitott maradt a „természetes" szegmentációs pontok megkeresésének és figyelembe vételének a kérdése. Ez jelentősen növelheti ennek a függvénynek a pontosságát és hatékonyságát. Ez viszont szintén komoly ötleteket igénylő részprobléma.

A programból teljes egészében kimaradt az utólagos ellenőrzési rész. Ez előreláthatólag nem lesz jelentős kihatással a program futására (nem lesz sokkal lassúbb), viszont jelentősen növelheti a pontos felismerés arányát. Utólagos ellenőrzés alatt olyan dolgokat lehet megvizsgálni, hogy az eredmény valóban az-e, ami oda van írva. Olyan jellegű vizsgálatokra gondolok itt, amelyek viszonylag könnyen elvégezhetőek, és érdemes elvégezni az egymásra hasonlító betűk megkülönböztetésére. Például a `Q` betűt igen gyakran hibásan `O` betűként ismeri fel. Tehát ha az eredmény `O`, érdemes utólag megvizsgálni azt, hogy ez valóban az-e, vagypedig `Q`. Ezt a vizsgálatot viszonylag könnyen el lehet végezni.

A szűrésre is ki kellene fejleszteni egy „intelligens" módszert. Ugyanis ha túl vékony a betű, akkor a szűrés következtében szétszakadhat, másrészt viszont ha a lényeges üregek túl kicsik, akkor azok eltűnhetnek. Ráadásul elég gyakran egyáltalán nincs szükség szűrésre, mivel a betűk elég „szépek" ahhoz, hogy szűrés nélküli feldolgozásra alkalmasak legyenek.

Hosszabb távú fejlszetési stratégia lehet még a karakterkészlet kibővítése például a számokra, a kisbetűkre, a magyar ékezetes betűkre és az egyéb műveleti jelekre is.

A. Forrás

A.1. Deklarációk

```
// A programot Faragó Csaba készítette 1998-99-ben
// E-mail: h633185@stud.u-szeged.hu
// Web: http://www.cab.u-szeged.hu/~h633185/

#include <stdio.h>
#include <stdlib.h>
#define DEBUG 1
#define PRECIZ

const int betumagassag      = 106;
const int betuszelesseg     = 47;
const int sorokszama        = 20;
const int betukszama        = 26;
const int felsotav          = 50;
const int iteraciokszama    = 5;
const int vizszinteseltolas = 108;
const float kilenges        = -0.5;
const int elsoszam          = 2;
const int masodikszam       = 2;
const int fajta             = 40;
const int mintasor          = 0;


typedef char betu[betumagassag][betuszelesseg];
typedef struct pontstruktura {
    int x,y,hany;
} pont;
typedef struct listastruktura {
    pont p; struct listastruktura *next;
} listatip;

typedef struct TBitMapFileHeader {
    unsigned char bfType[2];
    unsigned char bfSize[4];
    unsigned char bfReserved1[2];
    unsigned char bfReserved2[2];
    unsigned char bfOffBits[4];
} TBITMAPFILEHEADER;

typedef struct TBitMapInfoHeader {
    unsigned char biSize[4];
    unsigned char biWidth[4];
    unsigned char biHeight[4];
    unsigned char biPlanes[2];
```

```
    unsigned char biBitCount[2];
    unsigned char biCompression[4];
    unsigned char biSizeImage[4];
    unsigned char biXPelsPerMeter[4];
    unsigned char biYPelsPerMeter[4];
    unsigned char biClrUsed[4];
    unsigned char biClrImportant[4];
} TBITMAPINFOHEADER;

typedef struct TRGBQuad {
    unsigned char red;
    unsigned char green;
    unsigned char blue;
    unsigned char reserved;
} TRGBQUAD;

short toShort(unsigned char a[2]) {
    return a[0]+256*a[1];
}

long toLong(unsigned char a[4]) {
    return a[0]+256*(a[1]+256*(a[2]+256*a[3]));
}

class verem {
    private:
        pont *tomb;
        // az adattombre mutato pointer
        int veremmutato;
        // az eppen aktualis veremmutato
        int veremmax;
        // a verem maximalis terhelhetosege
    public:
        verem();
        // konstruktor; 200 melysegu vermet keszit
        verem(int);
        // konstruktor; a parameterben megadott
        // melysegu vermet keszit
        ~verem();
        // destruktor; felszabaditja a vermet
        void push(pont);
        // beteves a verem tetejere
        void push(int,int);
        void pop(pont *);
        // kivetel a verem tetejerol
        int ures();
        // igaz, ha a verem ures, hamis kulonben
        int megtelt();
        // igaz, ha megtelt a verem, hamis kulonben
```

```
};

verem::verem(int x) {
    veremmutato=0;
    veremmax=x;
    tomb=(pont *)malloc(x*sizeof(pont));
}

verem::verem() {
    veremmutato=0;
    veremmax=200;
    tomb=(pont *)malloc(200*sizeof(pont));
}

verem::~verem() {
    free(tomb);
}

void verem::push(pont x) {
    if (veremmutato>=veremmax) {
        fprintf(stderr,"Megtelt a verem!\n");
        exit(3);
    } else
        tomb[veremmutato++]=x;
}

void verem::push(int x, int y) {
    pont p;
    p.x=x; p.y=y;
    push(p);
}

void verem::pop(pont *x) {
    if (veremmutato==0) {
        fprintf(stderr,"Ures a verem!\n");
        exit(4);
    } else
        *x=tomb[--veremmutato];
    return;
}

int verem::ures() {
    return !veremmutato;
}

int verem::megtelt() {
    return veremmutato==veremmax-1;
}
```

A.2. Előfeldolgozó rutinok

```
int bmp() {
    FILE *bitmap;
    TBITMAPFILEHEADER fileheader;
    TBITMAPINFOHEADER infoheader;
    TRGBQUAD *paletta;
    unsigned char *kep;

    bitmap=fopen("csaba1.bmp","r");
    if (!bitmap) {
        printf("Nem talalhato a fajl!\n");
        return 1;
    }
    fscanf(bitmap,"%14c",(char *)&fileheader);
    if (fileheader.bfType[0]!='B' ||
        fileheader.bfType[1]!='M') {
        printf("Ez nem BMP-fajl!\n");
        return 1;
    }
    fscanf(bitmap,"%40c",(char *)&infoheader);
    for (int i=40; i<toLong(infoheader.biSize); i++) {
        char temp;
        fscanf(bitmap,"%c",&temp);
    }
    if (toShort(infoheader.biBitCount)!=8) {
        printf("Ez nem 8 biten kodolt. ");
        printf("Ezzel most nem foglalkozom.\n");
        return 1;
    }
    if (toLong(infoheader.biCompression)!=0) {
        printf("Ez kodolt BMP kep. ");
        printf("Ezzel most nem foglalkozom.\n");
        return 1;
    }
    paletta=(TRGBQUAD *)malloc(toLong(infoheader.biClrUsed)*
                            sizeof(TRGBQUAD));
    for (int i=0; i<toLong(infoheader.biClrImportant
                  ?infoheader.biClrImportant
                  :infoheader.biClrUsed); i++)
        fscanf(bitmap,"%4c",(char *)&paletta[i]);

    long meret=toLong(infoheader.biSizeImage);
    kep=(unsigned char *)malloc(meret);
    if (!kep) {
        printf("Nincs eleg memoria!!!\n");
        return 1;
    }
```

```
    if (DEBUG) printf("A BMP fajl beolvasasa...\n");
    for (long i=0; i<meret;i++)
        fscanf(bitmap,"%c",&kep[i]);
    fclose(bitmap);
    if (DEBUG) printf("A BMP fajl feldolgozasa...\n");
    long xd=toLong(infoheader.biWidth);
    long yd=toLong(infoheader.biHeight);
    unsigned char bit[yd][xd];
    for (long i=0; i<yd; i++) {
        for (long j=0; j<xd; j++) {
            bit[i][j]=kep[i*(xd+(xd%4?(4-xd%4):0))+j];
        }
        if (DEBUG) if (!(i%10)) {
            printf("*");
            fflush(stdout);
        }
    }
    if (DEBUG) printf("\nA raw fajl mentese...\n");
    FILE *ki=fopen("raw","w");
    for (int i=yd-1; i>=0; i--) {
        for (int j=0; j<xd; j++) {
            int index=bit[i][j];
            if (paletta[index].red+
                paletta[index].green<350)
                fprintf(ki,"1");
            else
                fprintf(ki,"0");
        }
        fprintf(ki,"\n");
        if (DEBUG) if (!(i%10)) {
            printf("*");
            fflush(stdout);
        }
    }
    fclose(ki);
    free(kep);
    if (DEBUG) printf("\n");
    return 0;
}

int feldolgoz() {
    FILE *be=fopen("raw","r");
    char sor[1430]; //1425 darab egysegbol all egy sor
    char betu[betumagassag][betuszelesseg][betukszama];

    if (DEBUG) printf("A raw fajl feldolgozasa...\n");
    //a felso felesleges sorok ures beolvasasa
    for (int i=0; i<felsotav; i++)
        fscanf(be,"%s\n",(char *)sor);
```

```
    for (int i=0; i<sorokszama; i++) {
    //sorok szama
        for (int k=0; k<betumagassag; k++) {
        //egy sor betu beolvasasa
            fscanf(be,"%s",sor);
            for (int l=0; l<betukszama; l++) {
            //egy sorban ennyi betu van
                for (int m=0; m<betuszelesseg; m++) {
                //egy betu ilyen szeles
                    betu[k][m][l]=sor[vizszinteseltolas+
                                  int(l*betuszelesseg)+
                                  m+int(i*kilenges)];
                }
            }
        }

        for (int k=0; k<betukszama; k++) {
            if (DEBUG) {
                printf("*");
                fflush(stdout);
            }
            char fajlnev[8];
            fajlnev[0]=(i/10)+'0';
            fajlnev[1]=(i%10)+'0';
            fajlnev[2]='x';
            fajlnev[3]=k+'A';
            fajlnev[4]=0;
            FILE *ki=fopen(fajlnev,"w");
            if (!ki) {
                printf("IO-hiba\n");
                return 1;
            }
            for (int l=0; l<betumagassag; l++) {
                for (int m=0; m<betuszelesseg; m++)
                    fprintf(ki,"%c",betu[l][m][k]);
                fprintf(ki,"\n");
            }
            fclose(ki);
        }
    }
    if (DEBUG) printf("\n");
    fclose(be);
    return 0;
}
```

A.3. A szűrés rutinjai

```
void dilatacio(betu *x) {
    betu y;
    for (int i=0; i<betumagassag; i++)
        for (int j=0; j<betuszelesseg; j++)
            y[i][j]=(*x)[i][j];
    for (int i=0; i<betumagassag; i++)
    for (int j=0; j<betuszelesseg; j++) {
        if ((*x)[i][j]=='0') {
            if (i>0) if ((*x)[i-1][j]=='1') {
                y[i][j]='1';
                continue;
            }
            if (j>0) if ((*x)[i][j-1]=='1') {
                y[i][j]='1';
                continue;
            }
            if (i<betumagassag-1) if ((*x)[i+1][j]=='1') {
                y[i][j]='1';
                continue;
            }
            if (j<betuszelesseg-1) if ((*x)[i][j+1]=='1') {
                y[i][j]='1';
                continue;
            }
        }
    }
    for (int i=0; i<betumagassag; i++)
        for (int j=0; j<betuszelesseg; j++)
            (*x)[i][j]=y[i][j];
}

void erozio(betu *x) {
    betu y;
    for (int i=0; i<betumagassag; i++)
        for (int j=0; j<betuszelesseg; j++)
            y[i][j]=(*x)[i][j];
    for (int i=0; i<betumagassag; i++)
    for (int j=0; j<betuszelesseg; j++) {
        if ((*x)[i][j]=='1') {
            if (i==0) {
                y[i][j]='0';
                continue;
            }
            if (j==0) {
                y[i][j]='0';
                continue;
```

```
            }
            if (i==betumagassag-1) {
                y[i][j]='0';
                continue;
            }
            if (j==betuszelesseg-1) {
                y[i][j]='0';
                continue;
            }
            if (i>0) if ((*x)[i-1][j]=='0') {
                y[i][j]='0';
                continue;
            }
            if (j>0) if ((*x)[i][j-1]=='0') {
                y[i][j]='0';
                continue;
            }
            if (i<betumagassag-1) if ((*x)[i+1][j]=='0') {
                y[i][j]='0';
                continue;
            }
            if (j<betuszelesseg-1) if ((*x)[i][j+1]=='0') {
                y[i][j]='0';
                continue;
            }
        }
    }
    for (int i=0; i<betumagassag; i++)
        for (int j=0; j<betuszelesseg; j++)
            (*x)[i][j]=y[i][j];
}

int szures() {
    if (DEBUG) printf("Morfologiai szures...\n");
    for (int i=0; i<sorokszama; i++)
    for (int j=0; j<betukszama; j++) {
        char fajlnev[8];
        fajlnev[0]=i/10+'0';
        fajlnev[1]=i%10+'0';
        fajlnev[2]='x';
        fajlnev[3]=j+'A';
        fajlnev[4]=0;
        betu x;
        FILE *be=fopen(fajlnev,"r");
        if (!be) {
            printf("\nI/O error: %s\n",fajlnev);
            continue;
        }
        for (int k=0; k<betumagassag; k++) {
```

```
            for (int l=0; l<betuszelesseg; l++)
                fscanf(be,"%c",&x[k][l]);
            fscanf(be,"\n");
        }
        fclose(be);

        fajlnev[4]='s';
        fajlnev[5]=0;
        erozio(&x);
        dilatacio(&x);
        dilatacio(&x);
        erozio(&x);
        FILE *ki=fopen(fajlnev,"w");
        for (int k=0; k<betumagassag; k++) {
            for (int l=0; l<betuszelesseg; l++)
                fprintf(ki,"%c",x[k][l]);
            fprintf(ki,"\n");
        }
        fclose(ki);

        if (DEBUG) {
            printf("*");
            fflush(stdout);
        }
    }
    if (DEBUG) printf("\n");
    return 0;
}
```

A.4. A vékonyítás rutinjai

```
char thin_a[256] = {
    0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1,
    0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0,
```

```
    1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0  };

char thin_b[256] = {
    0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,
    0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
    1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
    1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0  };

char corr_a[256] = {
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
    1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
    1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
    1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
    1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0  };

char corr_b[256] = {
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1,
    0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0  };

char vegso[256] = {
    1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1,
    0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1,
    0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1,
    0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1,
    1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0,
    1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1,
    1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1,
    1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1,
    1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0,
    1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1,
    1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0  };

// ketmenetes vekonyito algoritmus
void vekonyit(betu *x) {
    int ok=1;
    betu y;
    for (int i=0; i<betuszelesseg; i++)
        (*x)[0][i]=(*x)[betumagassag-1][i]=
        (*x)[i][0]=(*x)[i][betuszelesseg-1]='0';
    if (betumagassag>betuszelesseg)
        for (int i=betuszelesseg; i<betumagassag; i++)
            (*x)[i][0]=(*x)[i][betuszelesseg-1]='0';
    else
        for (int i=betumagassag; i<betuszelesseg; i++)
            (*x)[i][0]=(*x)[betumagassag-1][i]='0';
    for (int i=0; i<betumagassag; i++)
        for (int j=0; j<betuszelesseg; j++)
            y[i][j]=(*x)[i][j];
    while (ok) {
        ok=0;
        //elso menet
        for (int i=1; i<betumagassag-1; i++) {
            for (int j=1; j<betuszelesseg-1; j++) {
                int index=128*((*x)[i-1][j-1]-'0')+
                          64*((*x)[i-1][ j ]-'0')+
                          32*((*x)[i-1][j+1]-'0')+
```

```
                    16*((*x)[ i ][j-1]-'0')+
                     8*((*x)[ i ][j+1]-'0')+
                     4*((*x)[i+1][j-1]-'0')+
                     2*((*x)[i+1][ j ]-'0')+
                       ((*x)[i+1][j+1]-'0');
            if ((*x)[i][j]=='1') if (thin_a[index]) {
                y[i][j]='0';
                ok=1;
            }
        }
    }
    for (int i=0; i<betumagassag; i++)
        for (int j=0; j<betuszelesseg; j++)
            (*x)[i][j]=y[i][j];
    //masodik menet
    for (int i=1; i<betumagassag-1; i++) {
        for (int j=1; j<betuszelesseg-1; j++) {
            int index=128*((*x)[i-1][j-1]-'0')+
                      64*((*x)[i-1][ j ]-'0')+
                      32*((*x)[i-1][j+1]-'0')+
                      16*((*x)[ i ][j-1]-'0')+
                       8*((*x)[ i ][j+1]-'0')+
                       4*((*x)[i+1][j-1]-'0')+
                       2*((*x)[i+1][ j ]-'0')+
                         ((*x)[i+1][j+1]-'0');
            if ((*x)[i][j]=='1') if (thin_b[index]) {
                y[i][j]='0';ok=1;
            }
        }
    }
    for (int i=0; i<betumagassag; i++)
        for (int j=0; j<betuszelesseg; j++)
            (*x)[i][j]=y[i][j];
}
//korrekcio, else menet
for (int i=1; i<betumagassag-1; i++) {
    for (int j=1; j<betuszelesseg; j++) {
            int index=128*((*x)[i-1][j-1]-'0')+
                   64*((*x)[i-1][ j ]-'0')+
                   32*((*x)[i-1][j+1]-'0')+
                   16*((*x)[ i ][j-1]-'0')+
                    8*((*x)[ i ][j+1]-'0')+
                    4*((*x)[i+1][j-1]-'0')+
                    2*((*x)[i+1][ j ]-'0')+
                      ((*x)[i+1][j+1]-'0');
        if ((*x)[i][j]=='1') if (corr_a[index])
            y[i][j]='0';
    }
}
```

```
    for (int i=0; i<betumagassag; i++)
        for (int j=0; j<betuszelesseg; j++)
            (*x)[i][j]=y[i][j];
    //masodik menet
    for (int i=1; i<betumagassag-1; i++) {
        for (int j=1; j<betuszelesseg-1; j++) {
                int index=128*((*x)[i-1][j-1]-'0')+
                       64*((*x)[i-1][ j ]-'0')+
                       32*((*x)[i-1][j+1]-'0')+
                       16*((*x)[ i ][j-1]-'0')+
                        8*((*x)[ i ][j+1]-'0')+
                        4*((*x)[i+1][j-1]-'0')+
                        2*((*x)[i+1][ j ]-'0')+
                          ((*x)[i+1][j+1]-'0');
                if ((*x)[i][j]=='1') if (corr_b[index])
                y[i][j]='0';
        }
    }
    for (int i=0; i<betumagassag; i++)
        for (int j=0; j<betuszelesseg; j++)
            (*x)[i][j]=y[i][j];
    //vegso menet
    for (int i=1; i<betumagassag-1; i++) {
        for (int j=1; j<betuszelesseg-1; j++) {
                int index=128*((*x)[i-1][j-1]-'0')+
                       64*((*x)[i-1][ j ]-'0')+
                       32*((*x)[i-1][j+1]-'0')+
                       16*((*x)[ i ][j-1]-'0')+
                        8*((*x)[ i ][j+1]-'0')+
                        4*((*x)[i+1][j-1]-'0')+
                        2*((*x)[i+1][ j ]-'0')+
                          ((*x)[i+1][j+1]-'0');
                if ((*x)[i][j]=='1') if (vegso[index])
                (*x)[i][j]='0';
        }
    }
}

int vekonyitas() {
    if (DEBUG) printf("Vekonyitas...\n");
    for (int i=0; i<sorokszama; i++)
    for (int j=0; j<betukszama; j++) {
        char fajlnev[8];
        fajlnev[0]=i/10+'0';
        fajlnev[1]=i%10+'0';
        fajlnev[2]='x';
        fajlnev[3]=j+'A';
        fajlnev[4]='s';
        fajlnev[5]=0;
```

```
        FILE *inp=fopen(fajlnev,"r");
        if (inp) {
            if (DEBUG) {
                printf("*");
                fflush(stdout);
            }
            betu x;
            for (int k=0; k<betumagassag; k++) {
                for (int l=0; l<betuszelesseg; l++)
                    fscanf(inp,"%c",&x[k][l]);
                fscanf(inp,"\n");
            }
            fclose(inp);
            vekonyit(&x);
            fajlnev[4]='v';
            FILE *outp=fopen(fajlnev,"w");
            for (int k=0; k<betumagassag; k++) {
                for (int l=0; l<betuszelesseg; l++) {
                    fprintf(outp,"%c",x[k][l]);
                }
                fprintf(outp,"\n");
            }
            fclose(outp);
        }
    }
    printf("\n");
    return 0;
}
```

A.5. A zavaró részek eltávolításának függvényei

```
char tablazat[256] = {
    1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1,
    1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1,
    1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1,
    1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0,
    1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1,
    1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
    1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0  };

void iteral(betu *b) {
    betu temp, eredeti;
    for (int i=0; i<betumagassag; i++)
    for (int j=0; j<betuszelesseg; j++) {
        temp[i][j]=eredeti[i][j]=(*b)[i][j];
    }
    for (int p=0; p<iteraciokszama; p++) {
        for (int i=1; i<betumagassag-1; i++)
            for (int j=1; j<betuszelesseg-1; j++)
                if (temp[i][j]=='1') {
                        int index=128*(temp[i-1][j-1]-'0')+
                               64*(temp[i-1][ j ]-'0')+
                               32*(temp[i-1][j+1]-'0')+
                               16*(temp[ i ][j-1]-'0')+
                                    8*(temp[ i ][j+1]-'0')+
                                    4*(temp[i+1][j-1]-'0')+
                                    2*(temp[i+1][ j ]-'0')+
                                   (temp[i+1][j+1]-'0');
                        if (tablazat[index]) (*b)[i][j]='0';
        }
        for (int i=0; i<betumagassag; i++)
            for (int j=0; j<betuszelesseg; j++)
                temp[i][j]=(*b)[i][j];
    }
    // most jon a visszaiteralas
    for (int p=0; p<iteraciokszama; p++) {
        for (int i=1; i<betumagassag-1; i++)
            for (int j=1; j<betuszelesseg-1; j++)
                if ((*b)[i][j]=='1') {
                        int index=128*((*b)[i-1][j-1]-'0')+
                               64*((*b)[i-1][ j ]-'0')+
                               32*((*b)[i-1][j+1]-'0')+
                               16*((*b)[ i ][j-1]-'0')+
                                8*((*b)[ i ][j+1]-'0')+
                                4*((*b)[i+1][j-1]-'0')+
                                2*((*b)[i+1][ j ]-'0')+
                                  ((*b)[i+1][j+1]-'0');
                    if (tablazat[index]) {
                        for (int k=-1; k<=1; k++)
                            for (int l=-1; l<=1; l++)
                                (*b)[i+k][j+l]=eredeti[i+k][j+l];
                    }
                }
    }
    return;
}
```

```
int lenyeges() {
    if (DEBUG) printf("A lenyegtelen reszek eltavolitasa...\n");
    for (int i=0; i<sorokszama; i++)
    for (int j=0; j<betukszama; j++) {
        char fajlnev[8];
        fajlnev[0] = i/10+'0';
        fajlnev[1] = i%10+'0';
        fajlnev[2] = 'x';
        fajlnev[3] = j+'A';
        fajlnev[4] = 'v';
        fajlnev[5] = 0;
        FILE *fajl = fopen(fajlnev,"r");
        if (!fajl) continue;
        if (DEBUG) {printf("*"); fflush(stdout);}
        betu b;
        for (int k=0; k<betumagassag; k++) {
            for (int l=0; l<betuszelesseg; l++) {
                fscanf(fajl,"%c",&b[k][l]);
            }
            fscanf(fajl,"\n");
        }
        fclose(fajl);
        iteral(&b);
        for (int k=0; k<betumagassag; k++)
            for (int l=0; l<betuszelesseg; l++)
                if (b[k][l]=='1') b[k][l]='2';
        for (int k=1; k<betumagassag-1; k++)
            for (int l=1; l<betuszelesseg-1; l++) {
                if ((b[k-1][l-1]-'0')+(b[k-1][l]-'0')+
                    (b[k-1][l+1]-'0')+(b[ k ][l-1]-'0')+
                    (b[ k ][l+1]-'0')+(b[k+1][l-1]-'0')+
                    (b[k+1][l]-'0')+(b[k+1][l+1]-'0')==0)
                    b[k][l]='0';
            }
        int x=betuszelesseg/2;
        int y=betumagassag/2;
        int irany=1;
        int ossz=1;
        int hatralevo=ossz;
        int ok=1;
        while ((b[y][x]!='2')&&ok) {
            switch (irany) {
                case 1 : if (x<1) {ok=0; break;}
                         x--;
                         hatralevo--;
                         if (!hatralevo) {
                             irany=2;
                             hatralevo=ossz;
                         }
```

```
                break;
        case 2 : if (y<1) {ok=0; break;}
                y--;
                hatralevo--;
                if (!hatralevo) {
                    irany=3;
                    ossz++;
                    hatralevo=ossz;
                }
                break;
        case 3 : if (x>=betuszelesseg) {
                    ok=0;
                    break;
                }
                x++;
                hatralevo--;
                if (!hatralevo) {
                    irany=4;
                    hatralevo=ossz;
                }
                break;
        default: if (y>=betumagassag) {
                    ok=0;
                    break;
                }
                y++;
                hatralevo--;
                if (!hatralevo) {
                    irany=1;
                    ossz++;
                    hatralevo=ossz;
                }
    } //switch
} //while
if (ok) {
    verem v;
    pont p;
    p.x=x;
    p.y=y;
    v.push(p);
    while (!v.ures()) {
        v.pop(&p);
        b[p.y][p.x]='1';
        for (int k=-1; k<=1; k++)
        for (int l=-1; l<=1; l++) {
            if (b[p.y+l][p.x+k]=='2')
                v.push(p.x+k,p.y+l);
        }
    }
```

```
        }
        for (int k=0; k<betumagassag; k++)
            for (int l=0; l<betuszelesseg; l++)
                if (b[k][l]=='2') b[k][l]='0';
        fajlnev[4]='l';
        FILE *out=fopen(fajlnev,"w");
        for (int k=0; k<betumagassag; k++) {
            for (int l=0; l<betuszelesseg; l++) {
                fprintf(out,"%c",b[k][l]);
            }
            fprintf(out,"\n");
        }
        fclose(out);
    }
    printf("\n");
    return 0;
}
```

A.6. A leíróvektort kialakító eljárás

```
int vektorkialakit() {
    printf("A leirovektor meghatarozasa...\n");
    for (int i=0; i<sorokszama; i++)
    for (int j=0; j<betukszama; j++) {
        char fajlnev[8];
        fajlnev[0] = i/10+'0';
        fajlnev[1] = i%10+'0';
        fajlnev[2] = 'x';
        fajlnev[3] = j+'A';
        fajlnev[4] = 'v';
        fajlnev[5] = 0;
        FILE *fajl = fopen(fajlnev,"r");
        if (!fajl) continue;
        if (DEBUG) {printf("*"); fflush(stdout);}
        betu x;
        for (int k=0; k<betumagassag; k++) {
            for (int l=0; l<betuszelesseg; l++)
                fscanf(fajl,"%c",&x[k][l]);
            fscanf(fajl,"\n");
        }
        fclose(fajl);
        verem v;
        //most kovetkezik a karakterisztikus
        //vektor mehatarozasa
        int vektor[65535];
        int hossz=0;
        listatip *lista=NULL, *aktualis=NULL, *elso=NULL;
```

```
pont pp;
for (int k=8; k<betumagassag-8; k++)
for (int l=1; l<betuszelesseg-2; l++)
if (x[k][l]=='1') {
    if (!lista) {
        pp.y=k; pp.x=l;
        for (int m=pp.y+1; m<
            (pp.y+4<betumagassag-1?pp.y+4:
             betumagassag-1); m++)
        for (int n=1; n<pp.x-5; n++) {
            if (x[m][n]=='1') {
                pp.y=m;
                pp.x=n;
                goto tovabb;
            }
        }
        for (int m=pp.y+1; m<
            (pp.y+10<betumagassag-1?pp.y+10:
             betumagassag-1); m++)
        for (int n=1; n<pp.x-12; n++) {
            if (x[m][n]=='1') {
                pp.y=m;
                pp.x=n;
                goto tovabb;
            }
        }
        tovabb:
        lista=(listatip *)malloc(sizeof(listatip));
        lista->p=pp;
        lista->next=NULL;
        elso=lista;
    }
    if ((k==pp.y)&&(l==pp.x)) continue;
    int szomszedokszama=-1;
    for (int m=-1; m<=1; m++)
    for (int n=-1; n<=1; n++)
        if (x[k+m][l+n]=='1') szomszedokszama++;
    if (szomszedokszama==1) {
        lista->next=(listatip *)malloc(sizeof(listatip));
        lista=lista->next;
        lista->p.y=k; lista->p.x=l;
        lista->next=aktualis;
    }
}
lista=elso;
while (1) {
    pont p;
    p=lista->p;
    aktualis=lista;
```

```
lista=lista->next;
free(aktualis);
v.push(p);
while (1) {
    vektor[hossz++]=0;
    while (!v.ures()) {
        v.pop(&p);
        x[p.y][p.x]='2';
        if (x[p.y][p.x-1]=='1') {
            v.push(p.x-1,p.y);
            vektor[hossz++]=8;
        }
        if (x[p.y+1][p.x-1]=='1') {
            v.push(p.x-1,p.y+1);
            vektor[hossz++]=7;
        }
        if (x[p.y+1][p.x]=='1') {
            v.push(p.x,p.y+1);
            vektor[hossz++]=6;
        }
        if (x[p.y+1][p.x+1]=='1') {
            v.push(p.x+1,p.y+1);
            vektor[hossz++]=5;
        }
        if (x[p.y][p.x+1]=='1') {
            v.push(p.x+1,p.y);
            vektor[hossz++]=4;
        }
        if (x[p.y-1][p.x+1]=='1') {
            v.push(p.x+1,p.y-1);
            vektor[hossz++]=3;
        }
        if (x[p.y-1][p.x]=='1') {
            v.push(p.x,p.y-1);
            vektor[hossz++]=2;
        }
        if (x[p.y-1][p.x-1]=='1') {
            v.push(p.x-1,p.y-1);
            vektor[hossz++]=1;
        }
    }
    for (int k=8; k<betumagassag-8; k++)
        for (int l=1; l<betuszelesseg-1; l++) {
            if (x[k][l]=='1') {
                p.y=k;
                p.x=l;
                v.push(p);
                goto ki;
            }
```

```
                    }
                break;
                ki: continue;
            }
            if (!lista) break;
            for (int k=0; k<betumagassag; k++)
                for (int l=0; l<betuszelesseg; l++)
                    if (x[k][l]=='2') x[k][l]='1';
            vektor[hossz++]=-1;
        }
        vektor[hossz++]=-1;
        fajlnev[4]='t';
        FILE *out=fopen(fajlnev,"w");
        for (int k=0; k<hossz; k++) {
            fprintf(out,"%d ",vektor[k]);
            if (vektor[k]==-1) fprintf(out,"\n");
        }
        fclose(out);
    }
    if (DEBUG) printf("\n");
    return 0;
}
```

A.7. A szemantikus felismerés függvényei

```
betu x;
int szamitasbajohet[betukszama];
int **minta;
int *vektor;
int mintatemp[255], vektortemp[255];

// ez a fuggveny a szakasz alakjat probalja megallapitani
int milyen(int szam) {
    pont kezdo,veg;
    kezdo.x=0;
    veg.x=0;
    int osszpont=0;
    int irany[4];
    for (int k=0; k<4; k++) irany[k]=0;
    for (int k=1; k<betumagassag-1; k++)
      for (int l=1; l<betuszelesseg-1; l++) {
        if (x[k][l]==szam) {
            osszpont++;
            int szomszedokszama=-1;
            for (int m=-1; m<=1; m++)
            for (int n=-1; n<=1; n++) {
                if (x[k+m][l+n]==szam) {
```

```
                szomszedokszama++;
                if ((m==0)&&(n==1)) irany[0]++;
                else if (m==1)
                    if (n==1) irany[1]++;
                    else if (n==0) irany[2]++;
                    else irany[3]++;
            }
        }
        if (szomszedokszama==1) {
            if (kezdo.x==0) kezdo.x=l, kezdo.y=k;
            else veg.x=l, veg.y=k;
        }
    }
}
// a trivialis esetek elvetese
if (osszpont<5) return 0;
if (!kezdo.x) return 0;
if (!veg.x) return 0;
// jo az irany?
pont temppont;
if (kezdo.x+kezdo.y>veg.x+veg.y) {
    temppont=kezdo;
    kezdo=veg;
    veg=temppont;
}
char vektor[osszpont];
//1. dim: hanyadik otod (5.: osszes);
//2. dim: irany: ENy, E, EK, Ny,
//<osszpont>, K, DNy, D, DK
int otod[6][10];
for (int k=0; k<6; k++)
    for (int l=0; l<9; l++)
        otod[k][l]=0;
temppont=kezdo;
int szamlalo=0;
int eltolas=fajta;
while (1) {
    x[temppont.y][temppont.x]=szam+eltolas;
    for (int m=-1; m<=1; m++)
    for (int n=-1; n<=1; n++)
        if (x[temppont.y+m][temppont.x+n]==szam) {
            otod[5][3*m+n+4]++;
            otod[5*(szamlalo+1)/(osszpont)][3*m+n+4]++;
            otod[5*(szamlalo+1)/(osszpont)][4]++;
            otod[5][4]++;
            vektor[szamlalo++]=3*m+n+4;
            temppont.x=temppont.x+n;
            temppont.y=temppont.y+m;
            goto ki;
```

```
        }
    break;
    ki: continue;
    break;
}

for (int k=0; k<betumagassag; k++)
    for (int l=0; l<betuszelesseg; l++)
        if (x[k][l]==szam+eltolas) x[k][l]=szam;

//eloszor a trivialis eseteket kezelem le
if (10*abs(kezdo.y-veg.y)+3>=8*(osszpont-2)) {
    if (abs(kezdo.x-veg.x)*2<=osszpont) {
        if (5*irany[2]>=2*osszpont-2) return 1;
        // fuggoleges
    } else if (((kezdo.x+3<veg.x)&&(kezdo.y+3<veg.y))||
               ((kezdo.x>veg.x+3)&&(kezdo.y>veg.y+3)))
        return 4; // DNY-EK
    else if (((kezdo.x>veg.x+3)&&(kezdo.y+3<veg.y))||
             ((kezdo.x+3<veg.x)&&(kezdo.y>veg.y+3)))
        return 3; // DK-ENY
    if (abs(kezdo.x-veg.x)*4>=osszpont)
        if ((kezdo.x-veg.x)*(kezdo.y-veg.y)>0) return 3;
        else return 4;
}
if (10*abs(kezdo.x-veg.x)+3>=8*(osszpont-2)) {
    if (abs(kezdo.y-veg.y)*2<=osszpont) {
        if (5*irany[0]>=2*osszpont-2) return 2;
        // vizszintes
    }
    else if (((kezdo.x+3<veg.x)&&(kezdo.y+3<veg.y))||
             ((kezdo.x>veg.x+3)&&(kezdo.y>veg.y+3)))
        return 4; // DNY-EK
    else if (((kezdo.x>veg.x+3)&&(kezdo.y+3<veg.y))||
             ((kezdo.x+3<veg.x)&&(kezdo.y>veg.y+3)))
        return 3; // DK-ENY
    if (abs(kezdo.y-veg.y)*4>=osszpont)
        if ((kezdo.x-veg.x)*(kezdo.y-veg.y)>0) return 3;
        else return 4;
}

// KAPU vagy KUPA?
if ((25*(otod[0][1]+otod[1][1])+
     20*(otod[0][2]+otod[1][2])+
     15*(otod[0][0]+otod[1][5])>=
     (osszpont-1)*7-2)&&
    (25*(otod[3][7]+otod[4][7])+
    20*(otod[3][8]+otod[4][8])+
    15*(otod[3][5]+otod[4][6])>=
```

```
    (osszpont-1)*7-2)&&
    (otod[3][2]+otod[3][5]+otod[3][8]>
    otod[3][0]+otod[3][3]+otod[3][6]))
    return 5; //kapu
if ((25*(otod[0][1]+otod[1][1])+
     20*(otod[0][0]+otod[1][0])+
     15*(otod[0][2]+otod[1][3])>=
     (osszpont-1)*7-2)&&
    (25*(otod[3][7]+otod[4][7])+
     20*(otod[3][6]+otod[4][6])+
     15*(otod[3][3]+otod[4][8])>=
     (osszpont-1)*7-2)&&
    (otod[3][2]+otod[3][5]+otod[3][8]<
     otod[3][0]+otod[3][3]+otod[3][6]))
    return 5; //kapu

if ((25*(otod[0][7]+otod[1][7])+
     20*(otod[0][8]+otod[1][8])+
     15*(otod[0][6]+otod[1][5])>=
     (osszpont-1)*7-2)&&
    (25*(otod[3][1]+otod[4][1])+
     20*(otod[3][2]+otod[4][2])+
     15*(otod[3][5]+otod[4][0])>=
     (osszpont-1)*7-2)&&
    (otod[3][2]+otod[3][5]+otod[3][8]>
     otod[3][0]+otod[3][3]+otod[3][6]))
    return 6; //kupa
if ((25*(otod[0][7]+otod[1][7])+
     20*(otod[0][6]+otod[1][6])+
     15*(otod[0][8]+otod[1][3])>=
     (osszpont-1)*7-2)&&
    (25*(otod[3][1]+otod[4][1])+
     20*(otod[3][0]+otod[4][0])+
     15*(otod[0][2]+otod[1][3])>=
     (osszpont-1)*7-2)&&
    (otod[3][2]+otod[3][5]+otod[3][8]<
     otod[3][0]+otod[3][3]+otod[3][6]))
    return 6; //kupa

// forditott C?
if ((25*(otod[0][5]+otod[1][5])+
     20*(otod[0][8]+otod[1][8])>=osszpont*7)&&
    (25*(otod[3][3]+otod[4][3])+
     20*(otod[3][6]+otod[4][6])>=osszpont*7)&&
    (otod[3][6]+otod[3][7]+otod[3][8]>
     otod[3][0]+otod[3][1]+otod[3][2]))
    return 7; //forditott C
if ((25*(otod[0][5]+otod[1][5])+
     20*(otod[0][2]+otod[1][2])>=osszpont*7)&&
```

```
    (25*(otod[3][3]+otod[4][3])+
     20*(otod[3][0]+otod[4][0])>=osszpont*7)&&
    (otod[3][6]+otod[3][7]+otod[3][8]<
     otod[3][0]+otod[3][1]+otod[3][2]))
    return 7; //forditott C

// C?
if ((25*(otod[0][3]+otod[1][3])+
     20*(otod[0][6]+otod[1][6])>=osszpont*7)&&
    (25*(otod[3][5]+otod[4][5])+
     20*(otod[3][8]+otod[4][8])>=osszpont*7)&&
    (otod[3][6]+otod[3][7]+otod[3][8]>
     otod[3][0]+otod[3][1]+otod[3][2]))
    return 8; // C
if ((25*(otod[0][3]+otod[1][3])+
     20*(otod[0][0]+otod[1][0])>=osszpont*7)&&
    (25*(otod[3][5]+otod[4][5])+
     20*(otod[3][2]+otod[4][2])>=osszpont*7)&&
    (otod[3][6]+otod[3][7]+otod[3][8]<
     otod[3][0]+otod[3][1]+otod[3][2]))
    return 8; // C

// kalap-le?
if (((otod[0][5]+otod[1][5])*25>osszpont*7)&&
    ((otod[3][7]+otod[4][7])*25+
     (otod[3][6]+otod[4][6]+otod[3][8]+otod[4][8])*15>
      osszpont*7))
    return 9;
if (((otod[0][5]+otod[1][5])*25+
     (otod[0][2]+otod[1][2]+otod[0][8]+otod[1][8])*15>
      osszpont*7)&&
    ((otod[3][7]+otod[4][7])*25>osszpont*7))
    return 9;
if (((otod[3][3]+otod[4][3])*25>osszpont*7)&&
    ((otod[0][1]+otod[1][1])*25+
     (otod[0][0]+otod[1][0]+otod[0][2]+otod[1][2])*15>
      osszpont*7))
    return 9;
if (((otod[3][3]+otod[4][3])*25+
     (otod[3][0]+otod[4][0]+otod[3][6]+otod[4][6])*15>
      osszpont*7)&&
    ((otod[0][1]+otod[1][1])*25>osszpont*7))
    return 9;

// fel-kalap?
if (((otod[0][1]+otod[1][1])*25+
     (otod[0][0]+otod[1][0]+otod[0][2]+otod[1][2])*15>
      osszpont*7)&&
    ((otod[3][5]+otod[4][5])*25>osszpont*7))
```

```
    return 10;
if (((otod[0][1]+otod[1][1])*25>osszpont*7)&&
    ((otod[3][5]+otod[4][5])*25+
     (otod[3][2]+otod[4][2]+otod[3][8]+otod[4][8])*15>
      osszpont*7))
    return 10;
if (((otod[3][7]+otod[4][7])*25+
     (otod[3][6]+otod[4][6]+otod[3][8]+otod[4][8])*15>
      osszpont*7)&&
    ((otod[0][3]+otod[1][3])*25>osszpont*7))
    return 10;
if (((otod[3][7]+otod[4][7])*25>osszpont*7)&&
    ((otod[0][3]+otod[1][3])*25+
     (otod[0][0]+otod[1][0]+otod[0][6]+otod[1][6])*15>
      osszpont*7))
    return 10;

// le-kalap?
if (((otod[0][7]+otod[1][7])*25+
     (otod[0][6]+otod[1][6]+otod[0][8]+otod[1][8])*15>
      osszpont*7)&&
    ((otod[3][5]+otod[4][5])*25>osszpont*7))
    return 11;
if (((otod[0][7]+otod[1][7])*25>osszpont*7)&&
    ((otod[3][5]+otod[4][5])*25+
     (otod[3][2]+otod[4][2]+otod[3][8]+otod[4][8])*15>
      osszpont*7))
    return 11;
if (((otod[3][1]+otod[4][1])*25+
     (otod[3][0]+otod[4][0]+otod[3][2]+otod[4][2])*15>
      osszpont*7)&&
    ((otod[0][3]+otod[1][3])*25>osszpont*7))
    return 11;
if (((otod[3][1]+otod[4][1])*25>osszpont*7)&&
    ((otod[0][3]+otod[1][3])*25+
     (otod[0][0]+otod[1][0]+otod[0][6]+otod[1][6])*15>
      osszpont*7))
    return 11;

// kalap-fel?
if (((otod[0][5]+otod[1][5])*25+
     (otod[0][2]+otod[1][2]+otod[0][8]+otod[1][8])*15>
      osszpont*7)&&
    ((otod[3][1]+otod[4][1])*25>osszpont*7))
    return 12;
if (((otod[0][5]+otod[1][5])*25>osszpont*7)&&
    ((otod[3][1]+otod[4][1])*25+
     (otod[3][0]+otod[4][0]+otod[3][2]+otod[4][2])*15>
      osszpont*7))
```

```
    return 12;
if (((otod[3][3]+otod[4][3])*25+
     (otod[3][0]+otod[4][0]+otod[3][6]+otod[4][6])*15>
      osszpont*7)&&
    ((otod[0][7]+otod[1][7])*25>osszpont*7))
    return 12;
if (((otod[3][3]+otod[4][3])*25>osszpont*7)&&
    ((otod[0][7]+otod[1][7])*25+
     (otod[0][6]+otod[1][6]+otod[0][8]+otod[1][8])*15>
      osszpont*7))
    return 12;

// J vagy forditott J?
if ((15*(otod[0][7]+otod[1][7]+otod[2][7]+otod[2][6])+
     10*(otod[0][6]+otod[1][6]+otod[2][3]+otod[0][8]+
         otod[1][8])>=(osszpont-1)*7-2)&&
    (25*(otod[3][6]+otod[3][3]+otod[3][0]+
         otod[4][3]+otod[4][0]+otod[4][1])+
     15*(otod[4][2]+otod[4][6]+otod[3][7]+otod[3][1])>=
        (osszpont-1)*7-2))
    return 13; //J
if ((15*(otod[2][1]+otod[3][1]+otod[4][1]+otod[2][2])+
     10*(otod[4][2]+otod[3][2]+otod[2][5]+otod[4][0]+
         otod[3][0])>=(osszpont-1)*7-2)&&
    (25*(otod[0][7]+otod[0][8]+otod[0][5]+
         otod[1][8]+otod[1][5]+otod[1][2])+
     15*(otod[0][2]+otod[0][6]+otod[1][1]+otod[1][7])>=
        (osszpont-1)*7-2))
    return 13; //J

if ((15*(otod[0][7]+otod[1][7]+otod[2][7]+otod[2][8])+
     10*(otod[0][8]+otod[1][8]+otod[2][5]+otod[0][6]+
         otod[1][6])>=(osszpont-1)*7-2)&&
    (25*(otod[3][5]+otod[3][2]+otod[3][8]+
         otod[4][5]+otod[4][2]+otod[4][1])+
     15*(otod[4][0]+otod[4][8]+otod[3][7]+otod[3][1])>=
        (osszpont-1)*7)-2)
    return 14; // forditott J
if ((15*(otod[2][1]+otod[3][1]+otod[4][1]+otod[2][0])+
     10*(otod[4][0]+otod[3][0]+otod[2][3]+otod[4][2]+
         otod[3][2])>=(osszpont-1)*7-2)&&
    (25*(otod[0][7]+otod[0][6]+otod[0][3]+
         otod[1][6]+otod[1][3]+otod[1][0])+
     15*(otod[0][0]+otod[0][8]+otod[1][1]+otod[1][7])>=
        (osszpont-1)*7)-2)
    return 14; // forditott J

// ismet: mas alaku (szabalyos) C?
if ((15*(otod[1][3]+otod[1][6]+otod[1][7]+
```

```
          otod[2][6]+otod[2][7]+otod[2][8]+
          otod[3][7]+otod[3][8]+otod[3][5])+
      10*(otod[1][0]+otod[3][2])>=osszpont*7)&&
      (50*(otod[0][1]+otod[0][0]+otod[0][3])+
       30*(otod[0][2]+otod[0][6])>=7*osszpont)&&
      (50*(otod[4][1]+otod[4][2]+otod[4][5])+
       30*(otod[4][0]+otod[4][8])>=7*osszpont))
    return 17; //C
if ((15*(otod[1][3]+otod[1][0]+otod[1][1]+
          otod[2][0]+otod[2][1]+otod[2][2]+
          otod[3][1]+otod[3][2]+otod[3][5])+
      10*(otod[1][6]+otod[3][8])>=osszpont*7)&&
      (50*(otod[0][3]+otod[0][6]+otod[0][7])+
       30*(otod[0][0]+otod[0][8])>=7*osszpont)&&
      (50*(otod[4][5]+otod[4][8]+otod[4][7])+
       30*(otod[4][6]+otod[4][2])>=7*osszpont))
    return 17; //C
// a fenti eset elforgatottjaival most nem foglalkozom

// most jonnek az S elforgatottjai.
// Csak a szabalyos S-sel foglalkozom
if ((8*(otod[0][0]+otod[0][3])+
      6*(otod[0][1]+otod[0][6])>osszpont)&&
     (8*(otod[1][7])+6*(otod[1][6]+otod[1][8])+
      4*(otod[1][5])>osszpont)&&
     (8*(otod[2][8]+otod[2][5])+
      6*(otod[2][7])>osszpont)&&
     (8*(otod[3][7])+6*(otod[3][6]+
         otod[3][8])>osszpont)&&
     (8*(otod[4][0]+otod[4][3])+
      6*(otod[4][6]+otod[4][1])>osszpont))
    return 21;
if ((8*(otod[0][5]+otod[0][8])+
      6*(otod[0][7]+otod[0][2])>osszpont)&&
     (8*(otod[1][1])+
      6*(otod[1][2]+otod[1][0])>osszpont)&&
     (8*(otod[2][0]+otod[2][3])+
      6*(otod[2][1])>osszpont)&&
     (8*(otod[3][1])+6*(otod[3][2]+otod[3][0])+
      4*(otod[3][3])>osszpont)&&
     (8*(otod[4][8]+otod[4][5])+
      6*(otod[4][2]+otod[4][7])>osszpont))
    return 21;

// "majdnem kor" vizsgalata
if (osszpont>10) {
    int jelzo=0, resz=osszpont;
    for (int k=0; k<9; k++) if (k!=4)
        jelzo+=abs(resz-otod[5][k]*8);
```

```
        jelzo/=8;
        if (4*jelzo<=3*osszpont) return fajta-1;
    }

    return -1;
}

char megprobalfelismerni(int i, int j) {
    char megoldas=0;
    char fajlnev[8];
    fajlnev[0]=i/10+'0';
    fajlnev[1]=i%10+'0';
    fajlnev[2]='x';
    fajlnev[3]=j+'A';
    fajlnev[4]='l';
    fajlnev[5]=0;
    FILE *fajl=fopen(fajlnev,"r");
    if (!fajl) return '!';
    for (int k=0; k<betumagassag; k++) {
        for (int l=0; l<betuszelesseg; l++)
            fscanf(fajl,"%c",&x[k][l]);
        fscanf(fajl,"\n");
    }
    fclose(fajl);
    int vegpontok=0; int haromszomszed=0;
    pont lenyegespontok[25];
    int lenyegespontokszama=0;
    int indulox=0, induloy=0;
    for (int k=1; k<betumagassag-1; k++)
    for (int l=1; l<betuszelesseg-1; l++)
        if (x[k][l]=='1') {
            int szomszedokszama=-1;
            for (int m=-1; m<=1; m++)
                for (int n=-1; n<=1; n++)
                    if (x[k+m][l+n]=='1')
                        szomszedokszama++;
            if (szomszedokszama!=2) {
                induloy=k,
                indulox=l;
                goto tovabb;
            }
        }
tovabb:
    int darab[255];
    pont sulypont[255];
    if (indulox) {
        int szamlalo='2';
        pont p;
        p.x=indulox;
```

```
p.y=induloy;
verem v;
v.push(p);
darab[0]=0;
darab[1]=szamlalo-'0';
darab[szamlalo-'0']=0;
sulypont[szamlalo-'0'].x=sulypont[szamlalo-'0'].y=0;
while (!v.ures()) {
    v.pop(&p);
    x[p.y][p.x]=szamlalo;
    if (szamlalo-'0'<255) {
        darab[szamlalo-'0']++;
        sulypont[szamlalo-'0'].x+=p.x;
        sulypont[szamlalo-'0'].y+=p.y;
    }
    int szomszedokszama=0;
    for (int k=-1; k<=1; k++)
        for (int l=-1; l<=1; l++)
            if ((k||l)&&(x[p.y+k][p.x+l]=='1')) {
                szomszedokszama++;
                if (x[p.y+k][p.x+l]=='1') {
                    v.push(p.x+l, p.y+k);
                    x[p.y+k][p.x+l]=1;
                }
            }
    if (szomszedokszama!=1) {
        if (szamlalo-'0'<254)
        if (darab[szamlalo-'0']>0) {
            sulypont[szamlalo-'0'].x/=
                darab[szamlalo-'0'];
            sulypont[szamlalo-'0'].y/=
                darab[szamlalo-'0'];
                sulypont[szamlalo-'0'+1].x=
                sulypont[szamlalo-'0'+1].y=0;
        }
        szamlalo++;
        if (szamlalo-'0'<255)
            darab[szamlalo-'0']=0,
            darab[1]++;
    }
}

fajlnev[4]='q';
FILE *outp=fopen(fajlnev,"w");
for (int k=0; k<betumagassag; k++) {
    for (int l=0; l<betuszelesseg; l++) {
        fprintf(outp,"%c",x[k][l]);
    }
    fprintf(outp,"\n");
```

```
    }
    fclose(outp);
}
if (darab[1]>200) return 1;
if (fajta>200) return 1;
int alakok[255]; int darabszam[255];
for (int k=0; k<fajta; k++) darabszam[k]=0;
for (int k=2; k<darab[1]; k++) {
    int temp=milyen(k+'0');
    alakok[k]=temp;
    if (temp>0) darabszam[temp]++, darab[0]++;
    else if (temp==-1) darabszam[0]++;
}
if (darabszam[0]) return megoldas;
for (int k=1; k<betumagassag-1; k++) {
    for (int l=1; l<betuszelesseg-1; l++)
    if (x[k][l]!='0') {
        int szomszedokszama=-1;
        for (int m=-1; m<=1; m++)
            for (int n=-1; n<=1; n++)
                if (x[k+m][l+n]!='0')
                    szomszedokszama++;
        if (szomszedokszama>3)
            szomszedokszama=3;
        if ((szomszedokszama!=2) &&
            (lenyegespontokszama<25)) {
            int ok=1;
            for (int m=lenyegespontokszama-1;
                     m>=0; m--) {
                if (abs(lenyegespontok[m].y-k)+
                    abs(lenyegespontok[m].x-l)+
                    abs(lenyegespontok[m].hany-
                        szomszedokszama)<=2) {
                    ok=0;
                    break;
                }
            }
            if (!ok) {
                continue;
            }
            lenyegespontok[lenyegespontokszama].y=k;
            lenyegespontok[lenyegespontokszama].x=l;
            lenyegespontok[lenyegespontokszama].hany=
                szomszedokszama;
            lenyegespontokszama++;
            if (szomszedokszama==1)
                vegpontok++;
            if (szomszedokszama>=3)
                haromszomszed++;
```

```
        }
    }
}
// most jonnek a talalgatasok
int minoseg;
switch (vegpontok) {
    case (0): switch (haromszomszed) {
                case (0): //D,O
                          for (int k=0; k<betukszama; k++)
                              szamitasbajohet[k]=0;
                          szamitasbajohet['D'-'A']=1;
                          szamitasbajohet['O'-'A']=1;
                          megoldas='1';
                          break;
                case (1):
                          break;
                case (2): megoldas='b';
                          if ((darabszam[2]==1)&&
                              (darabszam[5]==1)&&
                              (darabszam[6]==1)) {
                              megoldas='B';
                              break;
                          }
                          break;
                case (3):
                          break;
                default:
                          break;
            }
            break;
    case (1):switch (haromszomszed) {
                case (0):
                          break;
                case (1): megoldas='p'; //P, ritkan D
                          if ((darab[0]==2)&&
                              (darabszam[fajta-1]==1)&&
                              (darabszam[1]==1)) {
                              megoldas='P';
                              break;
                          }
                          megoldas='O';
                          break;
                case (2):
                          break;
                case (3): megoldas=0;
                          break;
                default :
                          break;
            }
```

```
          break;
case (2):switch (haromszomszed) {
              case (0): minoseg=milyen('2');
                        switch (minoseg) {
                            case (1) : megoldas='I';
                                       break;
                            case (6) : //U vagy V
                                           for (int k=0;
                                       k<betukszama; k++)
                                           szamitasbajohet[k]=0;
                                       szamitasbajohet['U'-'A']=1;
                                       szamitasbajohet['V'-'A']=1;
                                       megoldas='1';
                                       break;
                            case (8) : megoldas='C';
                                       break;
                            case (11): megoldas='L';
                                       break;
                            case (13): megoldas='J';
                                       break;
                            case (17):
                                           for (int k=0;
                                       k<betukszama; k++)
                                           szamitasbajohet[k]=0;
                                       szamitasbajohet['C'-'A']=1;
                                       szamitasbajohet['G'-'A']=1;
                                       megoldas='1';
                                       break;
                            case (21): megoldas='S';
                                       break;
                            default: megoldas=0;
                        }
                        break;
              case (1): megoldas=0; // ritka, hibas P
                        break;
              case (2): megoldas=0; //A,Q,R,D,G, hibas P
                        if ((darab[0]==4)&&
                            (darabszam[1]+darabszam[3]+
                             darabszam[4]==2)&&
                            (darabszam[2]==1)&&
                            (darabszam[5]==1)) {
                            int hanyadik=1,
                                mennyi=0,
                                vizszintes=1;
                            for (int k=2; k<((darab[1]<255)?
                                 darab[1]:255); k++) {
                            if (2*sulypont[k].x+
                                sulypont[k].y>mennyi)
                                mennyi=sulypont[k].x*2+
```

```
                                        sulypont[k].y,
                                        hanyadik=k;
                                    if (alakok[k]==2)
                                        vizszintes=k;
                                }
                                for (int k=0; k<betukszama; k++)
                                    szamitasbajohet[k]=0;
                                szamitasbajohet['A'-'A']=1;
                                szamitasbajohet['R'-'A']=1;
                                megoldas='1';
                                break;
                            }
                            if ((darab[0]==3)&&
                                (darabszam[fajta-1]==1)&&
                                (darabszam[1]+darabszam[4]==2)) {
                                for (int k=0; k<betukszama; k++)
                                    szamitasbajohet[k]=0;
                                szamitasbajohet['Q'-'A']=1;
                                szamitasbajohet['R'-'A']=1;
                                megoldas='1';
                                break;
                            }
                            megoldas=0;
                            break;
                    case (3):
                            break;
                    default :
                            break;
                }
                break;
case (3):switch (haromszomszed) {
                    case (0):
                            break;
                    case (1): megoldas='e';
                            //E,F,G,T,V,Y,N,J, hibas L
                            if ((darab[0]==2)&&
                                (darabszam[2]+darabszam[3]==1)&&
                                (darabszam[10]+darabszam[16]==1)&&
                                (darabszam[11]+darabszam[6]+
                                darabszam[14]==1)) {
                                megoldas='E';
                                break;
                            }
                            if ((darab[0]==2)&&
                                (darabszam[10]==1)&&
                                (darabszam[1]==1)&&
                                (darabszam[2]==1)) {
                                megoldas='F';
                                break;
```

```
                        }
                        if ((darab[0]==3)&&
                            (darabszam[4]+
                             darabszam[2]==2)&&
                            (darabszam[1]==1)) {
                            megoldas='T';
                            break;
                        }
                        if ((darab[0]==3)&&
                            (darabszam[4]==1)&&
                            (darabszam[3]==2)) {
                            megoldas='Y';
                            break;
                        }
                        if ((darab[0]==3)&&
                            (darabszam[17]==1)&&
                            (darabszam[2]==2)) {
                            megoldas='G';
                            break;
                        }
                        megoldas=0;
                        break;
                case (2): megoldas='a'; //ritka A
                        megoldas=0;
                        break;
                case (3): megoldas='q';
                        //hibas A, R vagy Q
                        megoldas=0;
                        break;
                default :
                        break;
            }
            break;
case (4):switch (haromszomszed) {
                case (0):
                        break;
                case (1): megoldas='k';
                        // idealis K, Z vagy J
                        megoldas=0;
                        break;
                case (2): megoldas=0; // H,J,K,M,N,X,Z,W
                        if ((darab[0]==5)&&
                            (darabszam[1]==4)&&
                            (darabszam[2]==1)) {
                            int hosszusag[4],
                                sorszam=0;
                            for (int k=0; k<darab[1]; k++)
                                if (alakok[k]==1) {
                                    hosszusag[sorszam++]=
```

```
                    darab[k];
                if (sorszam>=4)
                    break;
            }
        if ((2*hosszusag[0]<=
             3*hosszusag[1])&&
            (2*hosszusag[0]<=
             3*hosszusag[2])&&
            (2*hosszusag[0]<=
             3*hosszusag[3])&&
            (2*hosszusag[1]<=
             3*hosszusag[0])&&
            (2*hosszusag[1]<=
             3*hosszusag[2])&&
            (2*hosszusag[1]<=
             3*hosszusag[3])&&
            (2*hosszusag[2]<=
             3*hosszusag[0])&&
            (2*hosszusag[2]<=
             3*hosszusag[1])&&
            (2*hosszusag[2]<=
             3*hosszusag[3])&&
            (2*hosszusag[3]<=
             3*hosszusag[1])&&
            (2*hosszusag[3]<=
             3*hosszusag[2])&&
            (2*hosszusag[3]<=
             3*hosszusag[3])) {
            megoldas='H';
            break;
        } else {
            for (int k=0;
            k<betukszama; k++)
                szamitasbajohet[k]=0;
            szamitasbajohet['M'-'A']=1;
            szamitasbajohet['H'-'A']=1;
            megoldas='1';
            break;
        }
    }
    if ((darab[0]==3)&&
        (darabszam[1]==2)&&
        (darabszam[3]==1)&&
        (darabszam[4]==1)) {
        megoldas='K';
        break;
    }
    if ((darab[0]==4)&&
        (darabszam[1]==3)&&
```

```
                                (darabszam[3]==1)&&
                                (darabszam[4]==1)) {
                                megoldas='K';
                                break;
                            }
                            if ((darab[0]==4)&&
                                (darabszam[3]==2)&&
                                (darabszam[4]==2)) {
                                megoldas='X';
                                break;
                            }
                            if ((darab[0]==2)&&
                                (darabszam[9]==1)&&
                                (darabszam[13]==1)) {
                                megoldas='J';
                                break;
                            }
                            if ((darab[0]==2)&&
                                (darabszam[9]+
                                 darabszam[15]==1)&&
                                 (darabszam[11]+
                                  darabszam[14]==1)) {
                                megoldas='Z';
                                break;
                            }
                            if ((darab[0]==4)&&
                                (darabszam[1]+
                                 darabszam[3]+
                                 darabszam[4]==3)&&
                                (darabszam[13]==1)&&
                                (darabszam[14]==1)) {
                                megoldas='W';
                                break;
                            }
                            megoldas=0;
                            break;
                    case (3): // A dupla hibaval
                            megoldas=0;
                            break;
                    default : // (4,4) A is elofordulhat
                            megoldas=0;
                            break;
                }
                break;
    case (5):switch (haromszomszed) {
                    case (0):
                            break;
                    case (1):
                            break;
```

```
                    case (2):
                              break;
                    case (3):
                              break;
                    default :
                              break;
                }
                break;
        case (6):switch (haromszomszed) {
                    case (0):
                              break;
                    case (1):
                              break;
                    case (2): megoldas=0; //W
                              break;
                    case (3):
                              break;
                    default :
                              break;
                }
                break;
        default: break;
    }
    return megoldas;
}
```

A.8. A statisztikus felismerés függvényei

```
// leghosszabb kozos reszsorozat
int lhkrs() { // mintatemp, vektortemp
    int matrix[255][255];
    for (int i=0; i<250; i++)
        for (int j=0; j<250; j++)
            matrix[i][j]=0;
    for (int i=mintatemp[0]; i>=0; i--) {
        for (int j=vektortemp[0]; j>=0; j--) {
            if (matrix[i+1][j]>matrix[i][j])
                matrix[i][j]=matrix[i+1][j];
            if (matrix[i][j+1]>matrix[i][j])
                matrix[i][j]=matrix[i][j+1];
            if (matrix[i][j]<(matrix[i+1][j+1]+
                (mintatemp[i]==vektortemp[j]))) {
                matrix[i][j]=matrix[i+1][j+1]+
                (mintatemp[i]==vektortemp[j]);
            }
        }
    }
```

```
    return matrix[0][0];
}

int felismer() {
    printf("Most jon a felismeres...\n");
    printf("     ABCDEFGHIJKLMNOPQRSTUVWXYZ\n\n");
    minta=(int **)malloc(betukszama*sizeof(int *));
    for (int i=0; i<betukszama; i++) {
        minta[i]=(int *)malloc(65535*sizeof(int));
    }
    vektor=(int *)malloc(65535*sizeof(int));
    for (int i=0; i<26; i++) {
        char fajlnev[8];
        fajlnev[0]='0';
        fajlnev[1]=mintasor+'0';
        fajlnev[2]='x';
        fajlnev[3]=i+'A';
        fajlnev[4]='t';
        fajlnev[5]=0;
        FILE *mintafajl=fopen(fajlnev,"r");
        if (!mintafajl) continue;
        for (int j=1; j<65530; j++) {
            if (feof(mintafajl)) break;
            minta[i][0]=j;
            int beolvas=0;
            fscanf(mintafajl,"%d ",&beolvas);
            minta[i][j]=beolvas;
        }
        fclose(mintafajl);
    }
    FILE *outp=fopen("megoldas","w");
    for (int i=0; i<sorokszama; i++) {
            printf("%3d: ",i);
        for (int j=0; j<betukszama; j++) {
            char eredmeny;
            int hossz;
            FILE *fajl;
            char kod=eredmeny=
                 megprobalfelismerni(i,j);
            if (((eredmeny<='Z')&&(eredmeny>='A'))) {
                //jaj de jo, megvan a megoldas
            } else {
                if (eredmeny) {
                    eredmeny=kod=1;
                } else {
                    eredmeny=kod=0;
                }
                char fajlnev[8];
                hossz=0;
```

```
fajlnev[0] = i/10+'0';
fajlnev[1] = i%10+'0';
fajlnev[2] = 'x';
fajlnev[3] = j+'A';
fajlnev[4] = 't';
    fajlnev[5] = 0;
fajl = fopen(fajlnev,"r");
if (!fajl) {
    fprintf(outp," ");
    continue;
}
vektor[0]=0;
for (int k=1; k<65530; k++) {
    if (feof(fajl)) break;
    vektor[0]=k;
    int beolvas=0;
    fscanf(fajl,"%d ",&beolvas);
    vektor[k]=beolvas;
}
fclose(fajl);
for (int k=0; k<betukszama; k++)
    if ((kod==0)||((kod==1)&&
        (szamitasbajohet[k]))) {
        int index1=0;
        int darab1=0;
            while ((index1<vektor[0])&&
               (darab1<elsoszam)) {
            darab1++;
            vektortemp[0]=0;
            while ((index1<65535)&&
                   (vektor[++index1]!=-1)) {
                if (vektortemp[0]>250)
                    break;
                vektortemp[++vektortemp[0]]=
                    vektor[index1];
            }
            int index2=0;
            int darab2=0;
            while ((index2<minta[k][0])&&
                   (darab2<masodikszam)) {
                darab2++;
                mintatemp[0]=0;
                while ((index2<65535)&&
                       (minta[k][++index2]!=-1)) {
                    if (mintatemp[0]>250)
                        break;
                    mintatemp[++mintatemp[0]]=
                        minta[k][index2];
                }
```

```
                                int jelenlegi=lhkrs();
                                jelenlegi=(10*jelenlegi/vektortemp[0])*
                                          (10*jelenlegi/mintatemp[0]);
                                if (jelenlegi>hossz) {
                                    hossz=jelenlegi;
                                    eredmeny=k+'A';
                                    if (hossz>=80) goto elore;
                                }
                            }
                        }
                    }
                elore: fprintf(outp,"%c",eredmeny);
                printf("%c",eredmeny); fflush(stdout);
            }
            fprintf(outp,"\n");
            printf("\n");
        }
        fclose(outp);
        return 0;
}
```

A.9. Statisztika készítése és főprogram

```
int statisztika() {
    printf("Felismeresi statisztikak...\n");
    char fajlnev[12];
    fajlnev[0]='m';
    fajlnev[1]='e';
    fajlnev[2]='g';
    fajlnev[3]='o';
    fajlnev[4]='l';
    fajlnev[5]='d';
    fajlnev[6]='a';
    fajlnev[7]='s';
    fajlnev[8]=0;
    fajlnev[9]=0;
    FILE *be=fopen(fajlnev,"r");
    if (!be) {
        printf("Hiba a %s fajl megnyitasakor.\n",fajlnev);
        return 1;
    }
    char betuk[sorokszama][betukszama];
    int darab[betukszama];
    int ossz=0;
    for (int i=0; i<betukszama; i++) darab[i]=0;
    for (int i=0; i<sorokszama; i++) {
```

```
        for (int j=0; j<betukszama; j++) {
            fscanf(be,"%c",&betuk[i][j]);
            if (betuk[i][j]==j+'A') {darab[j]++; ossz++;}
        }
        fscanf(be,"\n");
    }
    printf("A %d betubol %d-t talalt el. Ez %d szazalek. ",
            sorokszama*betukszama,ossz,
            100*ossz/(sorokszama*betukszama));
    printf("Reszletes statisztikak:\n");
    for (int i=0; i<betukszama; i++) printf("%c:%3d   ",i+'A',
            100*darab[i]/sorokszama);
    printf("\n");
    fclose(be);
    return 0;
}

int main() {
    int rossz;
    rossz=bmp();
    if (rossz) {
        printf("Hiba tortent a bmp beolvasasakor.\n");
        return 1;
    }
    rossz=feldolgoz();
    if (rossz) {
        printf("Hiba tortent a feldolgozaskor.\n");
        return 2;
    }
    rossz=szures();
    if (rossz) {
        printf("Hiba tortent a szureskor.\n");
        return 3;
    }
    rossz=vekonyitas();
    if (rossz) {
        printf("Hiba tortent a vekonyitaskor.\n");
        return 4;
    }
    rossz=lenyeges();
    if (rossz) {
        printf("Hiba tortent a lenyeges resz meghatarozasakor.\n");
        return 5;
    }
    rossz=vektorkialakit();
    if (rossz) {
        printf("Hiba tortent a vektor meghatarozasakor.\n");
        return 6;
    }
```

```
    rossz=felismer();
    if (rossz) {
        printf("Hiba tortent a felismereskor.\n");
        return 7;
    }
    rossz=statisztika();
    if (rossz) {
        printf("Hiba tortent a statisztika keszitesekor.\n");
        return 8;
    }
    return 0;
}
```

B. Köszönetnyilvánítás

A vizsgadolgozatom elkészítéséhez nyújtott segítségekért hálás köszönettel tartozom

- **Csirik János** egyetemi tanárnak, vizsgadolgozati témavezetőmnek, aki számos javaslattal segítette elő a vizsgadolgozat elkészítését.
- **Palágyi Kálmán** adjunktusnak, aki sok segítséget nyújtott a szűrő és a vékonyító algoritmusok megírásához.
- **Müller Helga** egyetemi hallgatónak, aki jelentős segítséget nyújtott azáltal, hogy az angol ábécét harmincszor leírta, feldolgozandó anyagot biztosítva számomra.
- A **Mars Network Kft.**-nek, ahol lehetővé tették számomra azt, hogy a mintát beszkenneljem.

Hivatkozások

[1] Fundamentals in Handwriting Recognition, Springer-Verlag, 1993

[2] Szűrös Csaba vizsgadolgozata, 1998