# Spregoval tanultam táblázatkezelést

Csernoch Mária[1], Simon Klaudia[2],
Brósch Éva[3], Kiss Éva[4]

[1] csernoch.maria@inf.unideb.hu
DE IK
[2] diusz.simon@gmail.com
DE IK
[3] evica0521@gmail.com
DE IK
[4] evi0307@gmail.com
DE IK

**Absztrakt.** Sprego (Spreadsheet Lego) a táblázatkezelő programok funkcionális nyelvét használva egy mély metakognitív megközelítésű problémamegoldó módszer, amely alkalmas a tanulók algoritmikus készségének, számítógépes gondolkodásának fejlesztésére, valamint tudást eredményez a hosszú távú memóriában. Mindezen tulajdonságait figyelembe véve alkalmas arra, hogy csökkentsük a hibás táblázatkezelői dokumentumok számát, a dokumentumok előállításához és értelmezéséhez szükséges időt, emberi és gépi erőforrás igényt. Ezen túl szolgálhat bevezető nyelvként a magas szintű programozási nyelvekhez, valamint végfelhasználók programozási nyelveként. A módszer azon alapszik, hogy minimális számú, általános célú függvényt tanítunk meg, valamint azt, hogy ezeket összeépítve többszintű függvényekké hogyan oldhatunk meg összetett problémákat. Tanulmányunk egy weblap táblázatát felhasználva mutat példákat arra, hogy a szövegkezelő függvényekkel hogyan tudunk táblázatkezelőben programozni.

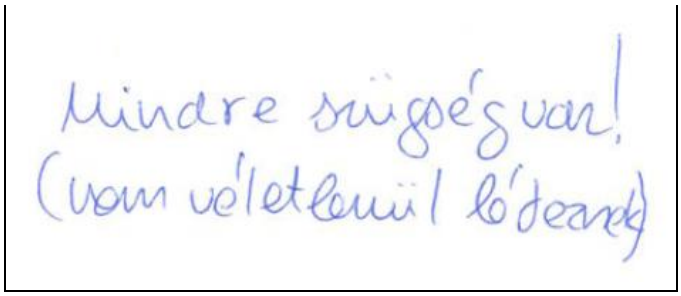
## 1. Bevezetés

Tanulmányunkban olyan feladatok megoldását mutatjuk be, amelyekkel szemléltethető, hogy táblázatkezelői környezetben klasszikus algoritmizálási problémák is megoldhatók, a nyelv alkalmas az algoritmikus készség, a számítógépes gondolkodás fejlesztésére [8], [9], [14], [15], [32].

A táblázatkezelő programok szinte kizárólagos használati módja a TAEW (Trial-and-Error Wizard) felületi metakognitív megközelítés [4], [5], [6], [10], [11], [12], [13], amely elsődlegesen a szoftvergyártó cégek szlogenje. Azt hirdetik és támogatják, hogy a táblázatkezelő programok használatához nem szükséges semmiféle ismeret, kattintgatásokkal, előre meg nem tervezett felületi navigációval el lehet jutni a végeredményhez. Ezt a szemléletet átvette az oktatási rendszerek többsége is, melynek szinte egyenes következménye a hibás dokumentumok rendkívül magas aránya – a mérési módszerektől függően ez 60–95%-ra tehető –, a dokumentumok előállításához és értelmezéséhez szükséges indokolatlanul magas emberi és gépi erőforrás igény [1], [2], [3], [10], [11], [12], [17], [18], [19], [20], [21], [22], [24], [25], [26]. Mindezek következménye, hogy a felhasználói dokumentumok TAEW módszerekkel történő barkácsolása [2], [3] óriási veszteségeket okoz mind a dokumentumok előállítóinak, mind az azokat értelmezőknek.

## 1.1. Sprego programozás

Az elterjedt TAEW típusú megközelítéssel szemben állítjuk, hogy táblázatkezelői környezetben fejleszthető az algoritmikus készség, a táblázatkezelő programok funkcionális nyelve szolgálhat bevezető nyelvként már általános és középiskolában is [7], [16], [23], [31]. Ehhez dolgoztuk és próbáltuk ki a Sprego (Spreadsheet Lego) programozást [6], [8], [9], [10], [11], [12]. Továbbá szeretnénk megmutatni, hogy egy táblázatkezelő program nem attól jó, mert rengeteg függvény van benne (1. ábra), szemben a közhiedelemmel.




**1. ábra:** Hallgatói vélemény, amely nagyon jól tükrözi a táblázatkezelő programok általános, ám hiba érzékeny megközelítését

A Sprego programozás lényege, hogy a létező legkevesebb általános célú függvényt használva hozzunk létre összetett függvényeket. A módszer egy tucat függvényt használ kiindulásként, valamint megengedi ezen készlet bővítését további általános célú függvényekkel (1. táblázat). Ennek megfelelően három Sprego csoportot hoztunk létre: Sprego 1 és 2 csoportok tartalmazzák a feltétlenül szükséges egy tucat függvényt, míg Sprego 3 a további opcionális függvényeket. Az egyszerű függvényekből létrehozott összetett függvényekkel már bonyolult problémák megoldása is lehetséges.

A Sprego programozás további előnye egyrészt, hogy nagyban támaszkodik a matematikából ismert függvény fogalomra [7], [8], [23], ugyanakkor az összetett függvények intenzív használatával megerősíti ezt a fogalmat, különös tekintettel az értelmezési tartomány és az értékkészlet közötti kapcsolatra. Másrészt, a magas szintű programozási nyelvekkel szemben, óriási előny a nyelv egyszerűsége, tehát a hangsúly nem a kódolás részletein van, hanem az algoritmus építésén és tesztelésén. Sprego programozási környezetben kiemelt fontosságú a nyelv egyszerűsége, tehát a hangsúly még inkább a feladat elemzésén, az algoritmus megépítésén, a kapott eredmények tesztelhetőségén és tesztelésén van.

Az előzőeken túl, a Sprego előnye még a két leggyakoribb táblázatkezelő, az MS Excel és az OpenOffice, LibreOffice Calc, valamennyi verziója közötti teljes kompatibilitás. Mivel a Sprego csak általános célú egyszerű függvényeket használ, ezért nincs szükség sem a programok, sem a verziók folyamatos ellenőrzésére. Az egyszerű függvények használata mentesíti továbbá a felhasználókat a probléma specifikus függvények használatától, amelyek következetlen argumentumlistái, nehezen érthető varázslói és súgói tovább növelik a dokumentumok hibaérzékenységét.

| Sprego 1 | Sprego 2 | Sprego 3 |
|---|---|---|
| SZUM() | INDEX() | KICSI() |
| ÁTLAG() | HOL.VAN() | NAGY() |
| MIN() | HIBÁS() | SOR() |
| MAX() | | OSZLOP() |
| BAL() | | ÉS() |
| JOBB() | | VAGY() |
| HOSSZ() | | NEM() |
| SZÖVEG.KERES() | | HELYETTE() |
| HA() | | ELTOLÁS() |
| | | TRANSZPONÁLÁS() |
| | | KEREKÍTÉS() |
| | | VÉL() |
| | | INT() |

**1. táblázat:** A Sprego függvények három csoportja

## 1.2. Táblázatkezelési problémák

A táblázatkezelőben történő programozás egy híd a végfelhasználói tevékenységek és a valódi programozás között. A már felsorolt előnyökön túl mindenképpen fontos megemlíteni, hogy a táblázatkezelő programok rendelkeznek azzal a tulajdonsággal, amivel több iskolai célú programozási nyelv nem; egy olyan eszköz, amelyre az iskola elhagyása után is szükség lesz, valamint már a tanulása során lehet valódi, autentikus problémák megoldására használni. Tehát van „értelme” annak, hogy megtanulják a használatát.

A tanulmányban bemutatott feladatok is egy valódi weblap táblázatát dolgozzák fel. A weblap konvertálása során keletkezett és a hibás sablonok használatából eredeztethető hibák javítását végezzük el (2. fejezet). A megoldások során táblázatba rendezve mutatjuk meg a függvények visszaadott értékét, amely értékek a hozzá külső függvény argumentumai, ezzel kapcsolatot teremtve az egymásba ágyazott függvények értelmezési tartománya és értékkészlete között.

## 1.3. Tömbképletek

### 1.3.1. Tömbképletek előnyei

A feladatokat többértékű tömbképletekkel oldjuk meg. Ezzel a módszerrel kiváltható a képletek másolása és a másolásból származó hibák [8], [9], [27], [28], [29], [31].

A tömbképlet egyik előnye a másolással szemben, hogy így csak egyetlen képlet keletkezik szemben a másolással előállított nagyszámú képlettel, míg a másik, hogy a tömbképletekkel el tudjuk kerülni a képletmásolást, ezzel együtt a másolásból származó hibákat is. A képletek másolásából származó hibák közül a leggyakoribbak:

- a rögzítésekből származó hibák
  - a rögzítés elhagyása vagy
  - túlzott használata
- a képlet módosítása nem az első előfordulásnál, ennek következménye, hogy
  - a módosítás nem történik meg az összes képletben
  - nem történik meg a módosított képlet átmásolása valamennyi előfordulásra.

#### 1.3.2. Többértékű tömbképletek létrehozása

A tömbképletek létrehozása többféle módon is megtörténhet, de talán az egyik legbiztonságosabb megoldás a következő:

- Létrehozzuk a tömbképletet a tömb legelső cellájában. A képlet fordítását és kiértékelését a Ctrl+Shift+Enter billentyűkombinációval indítjuk el. A kiértékelés eredményeként kiíratásra kerül a tömb első cellájában az első output.
- Kijelöljük a teljes tömböt, ahova az eredményeket szeretnénk kiíratni.
- Újra szerkeszthetővé tesszük a képletet.
- Ismét fordíttatjuk és kiértékeltetjük a képletet, tehát ismételten a Ctrl+Shift+Enter billentyűkombinációval zárjuk.

A Ctrl+Shift+Enter billentyűkombináció hatására megjelenik a képlet körül egy { } zárójelpár. Ezek a zárójelek nem gépelhetőek, csak és kizárólag a Ctrl+Shift+Enter billentyűkombinációval helyezhetők el.

A feladatokat megpróbáljuk segédoszlopok használata nélkül megoldani. Ahogy a tradicionális programozási nyelvekben is megszoktuk, a program írása során kell arról dönteni, hogy a helyfoglalással vagy inkább a futási idővel takarékoskodunk. A segédoszlopok használata ugyan növeli a futási időt, de a táblázat mérete lényegesen csökkenthető.

# 2. Sprego feladatok

A problémák megoldása CAAD (Computer Algorithmic and Debugging) típusú, mély metakognitív megközelítéssel történik, a tradicionális programozási környezetekben használt módszerek adaptálásával. Egy adott problémához, elemezzük magát a problémát, a rendelkezésre álló inputot, ezt követően megépítjük az algoritmust, kódoljuk azt, majd teszteljük az eredményt.

## 2.1. Minta – A táblázat szerkezete

A tanulmányban részletezett feladatokhoz a „List of states and territories of the United States” című weblap (http://en.wikipedia.org/wiki/List_of_states_and_territories_of_the_United_States), az USA államait és azokhoz tartozó adatokat felsoroló táblázatát használjuk. A táblázat kezdő és záró sorait mutatja a 2. ábra, majd ennek táblázatkezelő dokumentummá konvertált verzióját a 3. ábra.

States of the United States

| State | Abbr. | Capital | Largest city[A][7] | Statehood | Population (2013 est)[8] | Total area in mi² (km²)[B][9] | Land area in mi² (km²)[B][9] | Water area in mi² (km²)[B][9] | House seat(s) |
|---|---|---|---|---|---|---|---|---|---|
| Alabama | AL | Montgomery | Birmingham | December 14, 1819 | 4,833,722 | 52,420 (135,767) | 50,645 (131,170) | 1,775 (4,597) | 7 |
| Alaska | AK | Juneau | Anchorage | January 3, 1959 | 735,132 | 665,384 (1,723,337) | 570,641 (1,477,950) | 94,743 (245,383) | 1 |
| Arizona | AZ | Phoenix | Phoenix | February 14, 1912 | 6,626,624 | 113,990 (295,233) | 113,594 (294,207) | 396 (1,026) | 9 |
| Arkansas | AR | Little Rock | Little Rock | June 15, 1836 | 2,959,373 | 53,179 (137,733) | 52,035 (134,770) | 1,143 (2,960) | 4 |
| California | CA | Sacramento | Los Angeles | September 9, 1850 | 38,332,521 | 163,695 (423,968) | 155,779 (403,466) | 7,916 (20,502) | 53 |
| Colorado | CO | Denver | Denver | August 1, 1876 | 5,268,367 | 104,094 (269,602) | 103,642 (268,432) | 452 (1,171) | 7 |
| Connecticut | CT | Hartford | Bridgeport | January 9, 1788 | 3,596,080 | 5,543 (14,356) | 4,842 (12,541) | 701 (1,816) | 5 |
| Delaware | DE | Dover | Wilmington | December 7, 1787 | 925,749 | 2,489 (6,446) | 1,949 (5,048) | 540 (1,399) | 1 |
| Utah | UT | Salt Lake City | Salt Lake City | January 4, 1896 | 2,900,872 | 84,897 (219,882) | 82,170 (212,819) | 2,727 (7,063) | 4 |
| Vermont | VT | Montpelier | Burlington | March 4, 1791 | 626,630 | 9,616 (24,905) | 9,217 (23,872) | 400 (1,036) | 1 |
| Virginia[G] | VA | Richmond | Virginia Beach | June 25, 1788 | 8,260,405 | 42,775 (110,787) | 39,490 (102,279) | 3,285 (8,508) | 11 |
| Washington | WA | Olympia | Seattle | November 11, 1889 | 6,971,406 | 71,298 (184,661) | 66,456 (172,120) | 4,842 (12,541) | 10 |
| West Virginia | WV | Charleston | Charleston | June 20, 1863 | 1,854,304 | 24,230 (62,755) | 24,038 (62,258) | 192 (497) | 3 |
| Wisconsin | WI | Madison | Milwaukee | May 29, 1848 | 5,742,713 | 65,496 (169,634) | 54,158 (140,269) | 11,339 (29,368) | 8 |
| Wyoming | WY | Cheyenne | Cheyenne | July 10, 1890 | 582,658 | 97,813 (253,335) | 97,093 (251,470) | 720 (1,865) | 1 |

**2. ábra:** Az USA államok weblap táblázatának első és utolsó néhány sora

| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | State | Abbr. | Capital | Largest city | Statehood | Population | Total area in mi2 (km2) |
| 2 | Alabama | AL | Montgomery | Birmingham | December 14, 1819 | 4,833,722 | 7004524200000000000052,420 (135,767) |
| 3 | Alaska | AK | Juneau | Anchorage | January 3, 1959 | 735,132 | 7005665384000000000665,384 (1,723,337) |
| 16 | Iowa | IA | Des Moines | Des Moines | December 28, 1846 | 3,090,416 | 7004562730000000000056,273 (145,746) |
| 17 | Kansas | KS | Topeka | Wichita | January 29, 1861 | 2,893,957 | 7004822780000000000082,278 (213,099) |
| 18 | Kentucky[C] | KY | Frankfort | Louisville | June 1, 1792 | 4,395,295 | 7004404080000000000040,408 (104,656) |
| 19 | Louisiana | LA | Baton Rouge | New Orleans | April 30, 1812 | 4,625,470 | 7004523780000000000052,378 (135,658) |
| 20 | Maine | ME | Augusta | Portland | March 15, 1820 | 1,328,302 | 7004353800000000000035,380 (91,634) |
| 46 | Vermont | VT | Montpelier | Burlington | March 4, 1791 | 626,63 | 7003961600000000009,616 (24,905) |
| 47 | Virginia[G] | VA | Richmond | Virginia Beach | June 25, 1788 | 8,260,405 | 7004427750000000000042,775 (110,787) |
| 50 | Wisconsin | WI | Madison | Milwaukee | May 29, 1848 | 5,742,713 | 7004654960000000000065,496 (169,634) |
| 51 | Wyoming | WY | Cheyenne | Cheyenne | July 10, 1890 | 582,658 | 7004978130000000000097,813 (253,335) |

**3. ábra:** Az USA államok weblap konvertálásával létrehozott táblázat (válogatott sorok)

A táblázat oszlopai a következő adatokat tartalmazzák:

- A – államok neve, fölösleges karakterek eltávolítása szükséges (2.2. fejezet)
- B – államok nevének rövidítése, nincs teendő
- C – államok fővárosa, nincs teendő
- D államok legnagyobb városa, nincs teendő
- E – csatlakozás dátuma (Statehood), dátummá alakítás szükséges (2.4. fejezet)
- F – lakosság (Population), egész számmá alakítás szükséges (2.3. fejezet)
- G – terület $mi^2$ és $km^2$-ben megadva, az adatok szétválasztása és a fölösleges karakterek eltávolítása szükséges (2.5. fejezet)

## 2.2. Államok neve

Az államok nevei fölösleges karaktereket tartalmaznak – A oszlop (2. ábra és 3. ábra). Feladatunk ezen karakterek eltávolítása, tehát az államok nevének letisztázása.

- Minden állam neve előtt – a névtől balra – található egy szóköz.
- Néhány állam nevét követően található egy sztring, amely az alábbi tulajdonságokkal rendelkezik (2. ábra Virginia, 3. ábra Kentucky és Virginia államok):
  - három karakter hosszúságú,
  - az eredeti sztring jobb oldalán helyezkedik el,
  - szögletes zárójelbe van foglalva.

### 2.2.2. Algoritmus: szóközök eltávolítása

- A szóköztől jobbra az összes karaktert kivágjuk és ez lesz a visszaadott érték.
- A kivágott és visszaadott sztring hossza eggyel kevesebb, mint az eredeti sztring hossza.

### 2.2.3. Kódolás: szóközök eltávolítása

- Meghatározzuk az eredeti sztring hosszát (M1), amelyhez a HOSSZ() függvényt használjuk.
- Meghatározzuk az kivágandó sztring hosszát (M2).
- Az eredeti sztring jobb oldaláról kivágjuk a csonka, szóköz nélküli sztringet. Ez lesz a képlet visszaadott értéke (M3).

{=HOSSZ(A2:A51)} **(M1)**

{=HOSSZ(A2:A51)-1} **(M2)**

Az M2 képlet visszaadott értéke, outputja az új sztring hossza. Ez egy egész szám. Ez a szám lesz a JOBB() függvény egyik inputja, a második argumentuma. A függvények összeépítésével a JOBB() függvény teljes egészében magába foglalja a HOSSZ() függvényt (M3). A feladat megoldása során keletkezett visszaadott értékeket a 2a–2b. táblázatok tartalmazzák: az első oszlop (rózsaszín) a kiindulási értékeket, míg az utolsó (zöld) a teljes képlet visszaadott értékét.

{=JOBB(A2:A51;HOSSZ(A2:A51)-1)} **(M3)**

### 2.2.4. Algoritmus: államok neve utáni karakterek eltávolítás

- Az államnevek hosszának meghatározása, mivel a visszaadott sztring hossza hárommal lesz kevesebb, mint a szóköz nélküli sztring vagy néggyel, mint az eredeti sztring.
- A felesleges karakterek nélküli sztring kivágása a szóközökkel csonkított sztringből.

### 2.2.5. Kódolás: államok neve utáni karakterek eltávolítás

- Az államnevek hosszának meghatározásához használhatjuk a szóközmentes (M4 majd M5) vagy az eredeti sztringet is (M6).
- A szóközmentes sztring bal széléről kivágjuk az állam nevét a BAL() függvénnyel (M7)

| Input | M1 | M2 | M3 | M6 | M7 |
|---|---|---|---|---|---|
| Alabama | 8 | 7 | Alabama | 4 | Alab |
| Alaska | 7 | 6 | Alaska | 3 | Ala |
| Iowa | 5 | 4 | Iowa | 1 | I |
| Kansas | 7 | 6 | Kansas | 3 | Kan |
| Kentucky[C] | 12 | 11 | Kentucky[C] | 8 | Kentucky |
| Louisiana | 10 | 9 | Louisiana | 6 | Louisi |
| Maine | 6 | 5 | Maine | 2 | Ma |
| Vermont | 8 | 7 | Vermont | 4 | Verm |
| Virginia[G] | 12 | 11 | Virginia[G] | 8 | Virginia |
| Wisconsin | 10 | 9 | Wisconsin | 6 | Wiscon |
| Wyoming | 8 | 7 | Wyoming | 4 | Wyom |

**2a. táblázat:** Az államok nevének tisztázásakor keletkezett visszaadott értékek

| M8 | M9 | M10 | M11 |
|---|---|---|---|
| #ÉRTÉK! | IGAZ | Alabama | Alabama |
| #ÉRTÉK! | IGAZ | Alaska | Alaska |
| #ÉRTÉK! | IGAZ | Iowa | Iowa |
| #ÉRTÉK! | IGAZ | Kansas | Kansas |
| 10 | HAMIS | HAMIS | Kentucky |
| #ÉRTÉK! | IGAZ | Louisiana | Louisiana |
| #ÉRTÉK! | IGAZ | Maine | Maine |
| #ÉRTÉK! | IGAZ | Vermont | Vermont |
| 10 | HAMIS | HAMIS | Virginia |
| #ÉRTÉK! | IGAZ | Wisconsin | Wyoming |
| #ÉRTÉK! | IGAZ | Wyoming | Alabama |

**2b. táblázat:** Az államok nevének tisztázásakor keletkezett visszaadott értékek

{=HOSSZ(JOBB(A2:A51;HOSSZ(A2:A51)-1))} **(M4)**

{=HOSSZ(JOBB(A2:A51;HOSSZ(A2:A51)-1))-3} **(M5)**

{=HOSSZ(A2:A51)-4} **(M6)**

A BAL() függvény első argumentuma a szóközmentes sztring, míg a második az állam nevének a hossza, ami az M5 vagy az M6 képletből visszaadott érték (M7).

{=BAL(JOBB(A2:A51;HOSSZ(A2:A51)-1);HOSSZ(A2:A51)-4)} **(M7)**

### 2.2.6. Algoritmus: szétválogatás a fölösleges karakterek alapján

A gond azonban ott van, hogy a fölösleges három karakter csak alkalmanként jelenik meg az államok neve után. Tehát gondoskodnunk kell arról, hogy megkülönböztessük a kétféle sztringet az alapján, hogy van-e a végén fölösleges három karakter vagy nincs.

- Két eset lehetséges attól függően, hogy van fölösleges karakter a sztring végén vagy nincs.
- Csak abban az esetben kell a karakterek levágását elvégezni, ha létezik a három fölösleges karakter, egyébként a szóközmentes sztringet kell visszaadni.

### 2.2.7. Kódolás: szétválogatás a fölösleges karakterek alapján

- Megnézzük, hogy van-e fölösleges három karakter a sztring végén. Ezt megtehetjük úgy, hogy rákeresünk valamelyik szögletes zárójelre a SZÖVEG.KERES() függvénnyel (M8).
- Megnézzük, hogy a SZÖVEG.KERES() függvénynek mi a visszaadott értéke. Ha megtalálta a zárójelet, akkor a függvény egy egész számmal tér vissza, a zárójel pozíciójával. Ha nem talált a függvény szögletes zárójelet, akkor hibaüzenettel tér vissza.
- Leellenőrizzük a SZÖVEG.KERES() függvény visszaadott értékét a HIBÁS() függvénnyel. A HIBÁS() függvény két értékkel térhet vissza: IGAZ vagy HAMIS, attól függően, hogy hibával jött vissza a SZÖVEG.KERES() függvény vagy sem (M9).
- A HIBÁS() függvény outputjától függően kiíratjuk vagy a teljes szóközmentes sztringet (M10) vagy a szóközmentes és csonkított sztringet (M11).

Ennél a feladatrésznél is érdemes a szögletes zárójelet az eredeti rövid sztringben keresni, tehát átmenetileg kimásolhatjuk a csonkított sztring képletét egy cellába.

{=SZÖVEG.KERES("[";A2:A51)} **(M8)**

{=HIBÁS(SZÖVEG.KERES("[";A2:A51))} **(M9)**

{=HA(HIBÁS(SZÖVEG.KERES("[";A2:A51));JOBB(A2:A51; HOSSZ(A2:A51)-1))} **(M10)**

{=HA(HIBÁS(SZÖVEG.KERES("[";A2:A51)); JOBB(A2:A51;HOSSZ(A2:A51)-1); BAL(JOBB(A2:A51;HOSSZ(A2:A51)-1);HOSSZ(A2:A51)-4))} **(M11)**

## 2.3. Lakosság (Population) egész számmá alakítása

Az automatikus típus felismerés a táblázatkezelő programok egy kényelmi szolgáltatása, és egyben egy olyan eszköz, amely kezdő felhasználók számára leegyszerűsíti a kódolás folyamatát. Ez az egyik magyarázat arra, hogy ezek a nyelvek használhatóak bevezető nyelvként. Hiszen, ebben az esetben nem kell a típus deklarációkkal időzni, a hangsúly nem a kódolás részletein van, hanem az algoritmuson. Az automatikus típus felismerés azonban nem működik 100%-os pontossággal. Az egyik gyakori hibaforrás a tizedes és az ezreselválasztó karakterek különböző haszná-

lata a különböző nyelvekben. Ebből ered az általunk választott táblázat konverziós hibáinak nagy része is a lakosság és a dátum számadatainál.

A lakosság eredeti egész számai az angol írásmód szerinti ezreselválasztó vessző karaktereket tartalmazzák (2. ábra). Az eredetileg egész számokat, a vessző számától függően a magyar Excel kétféleképpen értelmezi:

- egy vessző esetén valós számként – az eredeti érték ezred részének (3. ábra: Alaska, Vermont, Wyoming),
- egynél több vessző esetén sztringként (3. ábra: Alabama, Iowa, Kansas, Kentucky, Louisiana, Maine, Virginia, Wisconsin)

Az angol egész szám magyar valós számmá konvertálása során azonban adatvesztés is történhet, amennyiben az utolsó helyiértékeken 0 található. A feldolgozás során, a végrehajtás lépéseinek sorrendjénél, erre mindenképpen érdemes odafigyelni és elkerülni az adatvesztést.

### 2.3.1. Lakosság (Population) jellemzői

Az automatikus típus felismerés következetlenségeinek következtében az alábbi kétféle értéket kaptuk az eredeti egész számok helyett:

- Egy vesszőt tartalmazó valós számok, amelyek az eredeti érték ezred részei. Ebben az esetben a vessző tizedes vesszőként funkcionál.
- Több vesszőt tartalmazó sztringek.

A kétféle értéket szét kell választanunk és külön-külön elvégezni a konverziót, hogy visszakapjuk az eredeti értékeket. Érdemes először külön oszlopokban megcsinálni a kétféle konvertálást, majd ezt követően összemásolni egy képletbe, amely mindkét esetet tudja kezelni. A lépésenkénti visszaadott értékeket a 3. táblázat tartalmazza.

### 2.3.2. Algoritmus: valós számok visszaalakítása egész számmá

- A szám szorzása 1000-rel.

### 2.3.3. Kódolás: valós számok visszaalakítása egész számmá

{=F2:F51*1000} **(M12)**

### 2.3.4. Algoritmus: sztringek egész számmá alakítása

- Vesszők eltávolítása.
- Sztring számmá alakítása.

### 2.3.5. Kódolás: sztringek egész számmá alakítása

- A vesszők eltávolítását a sztringből a HELYETTE() függvénnyel érdemes elvégezni, ahol a vessző karaktert az üres sztringre cseréljük (M13).

A sztring a vessző eltávolítása után már csak számjegyeket tartalmaz, tehát egy szorzással számmá alakítható. A legegyszerűbb, ha a sztringet 1-gyel szorozzuk meg, hogy a szám értéke nem változzon (M14).

| Input | M12 | M13 | M14 | M15 | M16 | M17 |
|---|---|---|---|---|---|---|
| 4,833,722 | #ÉRTÉK! | 4833722 | 4833722 | IGAZ | 0 | 4833722 |
| 735,132 | 735132 | 735132 | 735132 | HAMIS | 735132 | 735132 |
| 3,090,416 | #ÉRTÉK! | 3090416 | 3090416 | IGAZ | 0 | 3090416 |
| 2,893,957 | #ÉRTÉK! | 2893957 | 2893957 | IGAZ | 0 | 2893957 |
| 4,395,295 | #ÉRTÉK! | 4395295 | 4395295 | IGAZ | 0 | 4395295 |
| 4,625,470 | #ÉRTÉK! | 4625470 | 4625470 | IGAZ | 0 | 4625470 |
| 1,328,302 | #ÉRTÉK! | 1328302 | 1328302 | IGAZ | 0 | 1328302 |
| 626,63 | 626630 | 62663 | 62663 | HAMIS | 626630 | 626630 |
| 8,260,405 | #ÉRTÉK! | 8260405 | 8260405 | IGAZ | 0 | 8260405 |
| 5,742,713 | #ÉRTÉK! | 5742713 | 5742713 | IGAZ | 0 | 5742713 |
| 582,658 | 582658 | 582658 | 582658 | HAMIS | 582658 | 582658 |

**3. táblázat:** A lakosság (Population) értékeinek visszaállításához használt képletek visszaadott értéke

{=HELYETTE(F2:F51;",";"")} **(M13)**

{=HELYETTE(F2:F51;",";"")*1} **(M14)**

### 2.3.6. Algoritmus: A kétféle adattípus szétválogatása

- Rákérdezünk arra, hogy a cella számot tartalmaz-e, tehát ellenőrizzük, hogy az adat szám vagy sztring.
- A választól függően vagy a szám vagy a sztring konverziót végezzük el.

### 2.3.7. Kódolás: A kétféle adattípus szétválogatása

- Megpróbáljuk szorzással számmá alakítani az eredeti értékeket. A visszakapott érték vagy egy szám vagy egy hibaüzenet. Ezzel már készen vagyunk (M12)
- A HIBÁS() függvénnyel lekezeljük a szorzás eredményeként kapott visszaadott értéket. A HIBÁS() függvén visszaadott értéke vagy IGAZ vagy HAMIS. A kódoláshoz érdemes kifelé bővíteni az M12 képletet (M15).
- Tovább bővíthetjük kifelé a képletet a HA() függvénnyel, amellyel fel tudjuk tenni a kérdést és rendelkezhetünk a visszaadott értékekről a válasz függvényébe. HA() függvény első argumentuma lesz a teljes HIBÁS() függvény, harmadik argumentuma, a HA() függvény hamis ága, pedig az átalakított szám. Ebben az esetben a HA() függvény igaz ágán 0 értéket kapunk visszaadott értékként (M16).
- Feltöltjük a HA() függvény igaz ágát a sztringből számmá konvertálás képlettel (M14). A teljes megoldást az M17 képlet mutatja.

{=HIBÁS(F2:F51*1000)} **(M15)**

{=HA(HIBÁS(F2:F51*1000);;F2:F51*1000)} **(M16)**

{=HA(HIBÁS(F2:F51*1000);HELYETTE(F2:F51;",";"")*1;F2:F51*1000)} **(M17)**

## 2.4. Dátum átalakítása

### 2.4.1. Dátumok jellemzői

A táblázatban használt aktuális dátum formátumot a magyar Excel felismeri, van azonban két további probléma, ami miatt a sztring dátummá konvertálása nem történik meg automatikusan:

- A hónapok neve angolul szerepel az eredeti táblázatban.
  - Azokban a hónapokban, ahol a hónap angol és magyar neve megegyezik megtörténik az automatikus dátum felismerés (november, december),
  - egyébként nem.
- Az évek többsége, a táblázat tartalmának megfelelően, 1900 előttiek, az Excel alapértelmezésben, azonban, csak 1900-tól képes az évek kezelésére. (Az 1900 előtti évek kezelésére le kell tölteni az XDATE Add-In bővítést, ám ezzel itt nem kívánunk részletesen foglalkozni.)

Az angol hónapnevek lecseréléshez szükségünk van egy segédtáblára, amely egyik vektora tartalmazza a hónapok nevét angolul, míg a másik magyarul. Ezt a segédtáblát az A53:B64 tömbben helyeztük el.

A dátum konvertálása során keletkezett visszaadott értékeket a 4a–4b. táblázatok tartalmazzák.

### 2.4.2. Algoritmus: Angol nyelvű hónap nevek lecserélése magyarra

- Kivágjuk a dátum sztringből a hónapot. Ehhez ismernünk kell a hónapok nevének hosszát, ami eggyel kevesebb, mint az azt követő szóköz pozíciója.
- Az angol névhez megkeressük a magyar párját a segédtáblázatban.
- Az angol hónap nevet lecseréljük a magyar hónap nevekre.
- A sztringet számmá alakítjuk.

| Input | 1900-as évek (M2:M51) | M18 | M19 | M20 | M21 |
|---|---|---|---|---|---|
| 14.dec.19 | 14.dec.19 | #ÉRTÉK! | #ÉRTÉK! | #ÉRTÉK! | #ÉRTÉK! |
| January 3, 1959 | January 3, 1959 | 8 | 7 | January | 1 |
| December 28, 1846 | 28.dec.46 | #ÉRTÉK! | #ÉRTÉK! | #ÉRTÉK! | #ÉRTÉK! |
| January 29, 1861 | January 29, 1961 | 8 | 7 | January | 1 |
| June 1, 1792 | June 1, 1992 | 5 | 4 | June | 6 |
| April 30, 1812 | April 30, 1912 | 6 | 5 | April | 4 |
| March 15, 1820 | March 15, 1920 | 6 | 5 | March | 3 |
| March 4, 1791 | March 4, 1991 | 6 | 5 | March | 3 |
| June 25, 1788 | June 25, 1988 | 5 | 4 | June | 6 |
| May 29, 1848 | May 29, 1948 | 4 | 3 | May | 5 |
| July 10, 1890 | July 10, 1990 | 5 | 4 | July | 7 |

**4a. táblázat:** A dátum (Statehood) konvertálásához használt képletek visszaadott értéke

| M22 | M23 | M24 | cella formátum | M25 | M26 |
|---|---|---|---|---|---|
| #ÉRTÉK! | #ÉRTÉK! | #ÉRTÉK! | #ÉRTÉK! | IGAZ | 1919.12.14 |
| január | január 3, 1959 | 21553 | 1959.01.03 | HAMIS | 1959.01.03 |
| #ÉRTÉK! | #ÉRTÉK! | #ÉRTÉK! | #ÉRTÉK! | IGAZ | 1946.12.28 |
| január | január 29, 1961 | 22310 | 1961.01.29 | HAMIS | 1961.01.29 |
| június | június 1, 1992 | 33756 | 1992.06.01 | HAMIS | 1992.06.01 |
| április | április 30, 1912 | 4504 | 1912.04.30 | HAMIS | 1912.04.30 |
| március | március 15, 1920 | 7380 | 1920.03.15 | HAMIS | 1920.03.15 |
| március | március 4, 1991 | 33301 | 1991.03.04 | HAMIS | 1991.03.04 |
| június | június 25, 1988 | 32319 | 1988.06.25 | HAMIS | 1988.06.25 |
| május | május 29, 1948 | 17682 | 1948.05.29 | HAMIS | 1948.05.29 |
| július | július 10, 1990 | 33064 | 1990.07.10 | HAMIS | 1990.07.10 |

**4b. táblázat:** A dátum (Statehood) konvertálásához használt képletek visszaadott értéke

### 2.4.3. Kódolás: Angol nyelvű hónap nevek lecserélése magyarra

- A hónap nevét követő szóköz pozíciójának meghatározása a SZÖVEG.KERES() függvénnyel (M18).
- A hónapnév hosszának meghatározása (M19).

- A hónap kivágása a sztringből (M20).
- Az angol hónapnevek pozíciójának a meghatározását az A53:A64 vektorban a HOL.VAN() függvénnyel végezhetjük el (M21). A segédtáblába mind a 12 hónap nevet felvettük, így bármilyen két nyelv között működik a konverzió.
- A magyar hónapnevek kiíratása a B53:B64 vektorból az INDEX() függvénnyel történik (M22).
- Az angol hónapnevek lecserélését magyarra az eredeti sztringben a HELYETTE() függvénnyel végezzük el (M23). A HELYETTE() függvény argumentumai a következők:
  - eredeti sztring,
  - M20 képlettel kivágott angol hónapnév,
  - M22 képlettel kiíratott magyar hónapnevek.
- Dátum sztringek számmá alakítása (M24).

```
{=SZÖVEG.KERES(" ";M2:M51)}                                                    (M18)
```

```
{=SZÖVEG.KERES(" ";M2:M51)-1}                                                  (M19)
```

```
{=BAL(M2:M51;SZÖVEG.KERES(" ";M2:M51)-1)}                                      (M20)
```

```
{=HOL.VAN(BAL(M2:M51;SZÖVEG.KERES(" ";M2:M51)-1);A53:A64;0)}                   (M21)
```

```
{=INDEX(B53:B64;                                                               (M22)
HOL.VAN(BAL(M2:M51;SZÖVEG.KERES(" ";M2:M51)-1);A53:A64;0))}
```

```
{=HELYETTE(M2:M51;BAL(M2:M51;SZÖVEG.KERES(" ";M2:M51)-1);                      (M23)
INDEX(B53:B64;HOL.VAN(BAL(M2:M51;SZÖVEG.KERES(" ";M2:M51)-1);
A53:A64;0))))}
```

```
{=HELYETTE(M2:M51;BAL(M2:M51;SZÖVEG.KERES(" ";M2:M51)-1);                      (M24)
INDEX(B53:B64;HOL.VAN(BAL(M2:M51;SZÖVEG.KERES(" ";M2:M51)-1);
A53:A64;0)))*1}
```

### 2.4.4. Algoritmus: Dátum típusának ellenőrzése

Annak ellenőrzésére, hogy a sikerült-e dátum automatikus felismerése vagy sem használhatjuk az M18 képlet szóköz keresését.

- Rákeresünk a szóközre.
- Megnézzük, hogy hibával tért-e vissza a függvény vagy a szóköz pozíciójával.
- A hiba kezelése.

### 2.4.5. Kódolás: Dátum típusának ellenőrzése

- A hiba ellenőrzését a HIBÁS() függvénnyel el tudjuk végezni. Ha az eredeti dátum sztring, akkor a SZÖVEG.KERES() megtalálja a szóközt és visszaadja a pozícióját, tehát ezekben az esetekben a HIBÁS() függvény HAMIS értéket ad vissza. Ha már megtörtént az automatikus tí-

pus felismerés, akkor nincs szóköz a kifejezésben és a HIBÁS() függvény IGAZ értéket ad vissza (M25).

- A hiba kezelésére a HA() függvényt használhatjuk (M26)
  - a függvény kérdése az M25 képlet,
  - az igaz ágon az eredeti dátumot adjuk vissza,
  - a hamis ágon az M24 képlet visszaadott értékét.

```
{=HIBÁS(SZÖVEG.KERES(" ";M2:M51))}                                    (M25)
```

```
{=HA(HIBÁS(SZÖVEG.KERES(" ";M2:M51));                                 (M26)
M2:M51;
HELYETTE(M2:M51;BAL(M2:M51;SZÖVEG.KERES(" ";M2:M51)-1);
INDEX(B53:B64;HOL.VAN(BAL(M2:M51;SZÖVEG.KERES(" ";M2:M51)-1);
A53:A64;0)))*1)}
```

## 2.5. Államok (Total area in mi$^2$ (km$^2$)) területe

### 2.5.1. A területek jellemzői

Az államok területe több sebből is vérzik:

- Egy cellában tárolták a mi$^2$ és a km$^2$ értékeket, amiket szét kell választani.
- A G2:G51 vektor tartalmaz egy 19 karakter hosszúságú nem nyomtató karaktersorozatot, a mi$^2$ értékektől balra, amit szintén el kell távolítani.

A kiértékelés lépésenként visszaadott értékeit az 5a–5b. táblázatok tartalmazzák.

### 2.5.2. A km$^2$ jellemzői

- Az eredeti sztring jobb oldalán helyezkednek el az értékek.
- Az angol ezreselválasztó karaktert, a vesszőt tartalmazzák.
- Zárójelben vannak.

### 2.5.3. Algoritmus: A km$^2$ értékek kivágása

- A számok különböző hosszúságúak, de
- valamennyi a nyitó zárójeltől jobbra helyezkedik el,
- az eredeti sztring jobb szélén,
- minden számot még követ egy záró zárójel, amit el kell távolítani.

| Input | M27 | M28 | M29 |
|---|---|---|---|
| 7004524200000000000052,420 (135,767) | 35 | 27 | 8 |
| 70056653840000000000665,384 (1,723,337) | 38 | 28 | 10 |
| 7004562730000000000056,273 (145,746) | 35 | 27 | 8 |
| 7004822780000000000082,278 (213,099) | 35 | 27 | 8 |
| 7004404080000000000040,408 (104,656) | 35 | 27 | 8 |
| 7004523780000000000052,378 (135,658) | 35 | 27 | 8 |
| 7004353800000000000035,380 (91,634) | 34 | 27 | 7 |
| 7003961600000000000009,616 (24,905) | 33 | 26 | 7 |
| 7004427750000000000042,775 (110,787) | 35 | 27 | 8 |
| 7004654960000000000065,496 (169,634) | 35 | 27 | 8 |
| 7004978130000000000097,813 (253,335) | 35 | 27 | 8 |

**5a. táblázat:** A km$^2$ konvertálásához használt képletek visszaadott értéke

| M30 | M31 | M32 | M33 |
|---|---|---|---|
| 135,767) | 135,767 | 135767 | 135767 |
| 1,723,337) | 1,723,337 | 1723337 | 1723337 |
| 145,746) | 145,746 | 145746 | 145746 |
| 213,099) | 213,099 | 213099 | 213099 |
| 104,656) | 104,656 | 104656 | 104656 |
| 135,658) | 135,658 | 135658 | 135658 |
| 91,634) | 91,634 | 91634 | 91634 |
| 24,905) | 24,905 | 24905 | 24905 |
| 110,787) | 110,787 | 110787 | 110787 |
| 169,634) | 169,634 | 169634 | 169634 |
| 253,335) | 253,335 | 253335 | 253335 |

**5b. táblázat:** A km$^2$ konvertálásához használt képletek visszaadott értéke

### 2.5.4. Kódolás: A km$^2$ értékek kivágása

- A szám hosszának meghatározásához ismernünk kell az eredeti sztring hosszát (M27) és a nyitó zárójel pozícióját (M28), ezek különbsége fogja adni a szám hosszát (M29).
- Jobbról kivágunk ennyi karaktert (M30).

- Eltávolítjuk a záró zárójelet a sztring végéről, tehát a sztring bal szélét vágjuk ki és adjuk vissza. Ezt egy BAL() függvénnyel tehetjük meg (M31). A BAL() függvény argumentumai a következők:
  - a JOBB() függvénnyel visszakapott sztring,
  - a JOBB() függvénnyel visszakapott sztring hosszánál egy karakterrel rövidebb.
- A sztringből eltávolítjuk a vesszőket a HELYETTE() függvénnyel, ahol a vesszőt cseréljük le az üres sztringre (M32).
- A BAL() függvénnyel visszaadott sztringet számmá alakítjuk egy szorzással (M33).

{=HOSSZ(G2:G51)} **(M27)**

{=SZÖVEG.KERES("(";G2:G51)} **(M28)**

{=HOSSZ(G2:G51)-SZÖVEG.KERES("(";G2:G51)} **(M29)**

{=JOBB(G2:G51;HOSSZ(G2:G51)-SZÖVEG.KERES("(";G2:G51))} **(M30)**

{=BAL(JOBB(G2:G51;HOSSZ(G2:G51)-SZÖVEG.KERES("(";G2:G51)); **(M31)**
HOSSZ(JOBB(G2:G51;HOSSZ(G2:G51)-SZÖVEG.KERES("(";G2:G51)))-1)}

{=HELYETTE(BAL(JOBB(G2:G51; **(M32)**
HOSSZ(G2:G51)-SZÖVEG.KERES("(";G2:G51));HOSSZ(JOBB(G2:G51;
HOSSZ(G2:G51)-SZÖVEG.KERES("(";G2:G51)))-1);",";"")}

{=HELYETTE(BAL(JOBB(G2:G51; **(M33)**
HOSSZ(G2:G51)-SZÖVEG.KERES("(";G2:G51));HOSSZ(JOBB(G2:G51;
HOSSZ(G2:G51)-SZÖVEG.KERES("(";G2:G51)))-1);",";"")*1}

### 2.5.5. A $mi^2$ jellemzői

- Minden érték a 20. pozíción kezdődik.
- A nyitó zárójel előtt végződik 2 karakterrel.

A kiértékelés során kapott visszaadott értékeket, lépésenként az 5c. táblázat mutatja.

### 2.5.6. Algoritmus: A $mi^2$ értékek kivágása

- Az eredeti sztring jobb oldaláról a 20. karaktertől kezdődően kivágunk minden karaktert, $mi^2$ és $km^2$ értékeit is.
- Az így kapott sztring bal oldaláról kivágjuk a számot, a $mi^2$ értéket.
- Vesszőket eltávolítjuk.
- Végül, a sztringet számmá alakítjuk.

Az algoritmus két utolsó lépésének sorrendje kiemelt fontosságú. Amennyiben nem ebben a sorrendben végezzük el a vesszők eltávolítását és a számmá alakítást, ez egy vesszős számoknál, ha az utolsó karakter(ek) 0, akkor adatvesztés történik.

| M34 | M35 | M36 | M37 | M38 | M39 |
|---|---|---|---|---|---|
| 16 | 52,420 (135,767) | 6 | 52,420 | 52420 | 52420 |
| 19 | 665,384 (1,723,337) | 8 | 665,384 | 665384 | 665384 |
| 16 | 56,273 (145,746) | 6 | 56,273 | 56273 | 56273 |
| 16 | 82,278 (213,099) | 6 | 82,278 | 82278 | 82278 |
| 16 | 40,408 (104,656) | 6 | 40,408 | 40408 | 40408 |
| 16 | 52,378 (135,658) | 6 | 52,378 | 52378 | 52378 |
| 15 | 35,380 (91,634) | 5 | 35,38 | 3538 | 3538 |
| 14 | 9,616 (24,905) | 5 | 9,616 | 9616 | 9616 |
| 16 | 42,775 (110,787) | 6 | 42,775 | 42775 | 42775 |
| 16 | 65,496 (169,634) | 6 | 65,496 | 65496 | 65496 |
| 16 | 97,813 (253,335) | 6 | 97,813 | 97813 | 97813 |

**5c. táblázat:** A $mi^2$ konvertálásához használt képletek visszaadott értéke

### 2.5.7. Kódolás: A $mi^2$ értékek kivágása

- A JOBB() függvénnyel kivágjuk az eredeti sztring végén található két számot (M35). A JOBB() függvény argumentumai:
  - eredeti sztring,
  - az eredeti sztring hosszánál 19 karakterrel kevesebb.
- A BAL() függvénnyel levágjuk a rövidített sztring bal széléről a $mi^2$ értékeket (M37). A BAL() függvény argumentumai:
  - rövid sztring,
  - a szám hossza, amit megkapunk, ha kivonjuk a sztring hosszából a nyitó zárójel pozícióját és csökkentjük az értéket a szóköz és a nyitózárójel karakterek hosszával (a képlet rövidebb lesz, ha ennek az értéknek a meghatározásához az eredeti sztringet használjuk (M36)).
- Vesszők eltávolítását a HELYETTE() függvénnyel végezzük el, ahol a vesszőt az üres sztringre cseréljük (M38).
- Sztring számmá alakítása egy szorzással (M39).

{=HOSSZ(G2:G51)-19} **(M34)**

{=JOBB(G2:G51;HOSSZ(G2:G51)-19)} **(M35)**

{=HOSSZ(G2:G51)-SZÖVEG.KERES("(";G2:G51)-2} **(M36)**

{=BAL(JOBB(G2:G51;HOSSZ(G2:G51)-19); **(M37)**

HOSSZ(G2:G51)-SZÖVEG.KERES("(";G2:G51)-2)}

{=HELYETTE(BAL(JOBB(G2:G51;HOSSZ(G2:G51)-19); HOSSZ(G2:G51)-SZÖVEG.KERES("(";G2:G51)-2);",";"")} **(M38)**

{=HELYETTE(BAL(JOBB(G2:G51;HOSSZ(G2:G51)-19); HOSSZ(G2:G51)-SZÖVEG.KERES("(";G2:G51)-2);",";"")*1} **(M39)**

## 3. Összegzés

A feldolgozásra kiválasztott táblázaton keresztül a táblázatkezelő programok szövegkezelő függvényeinek használatára mutatunk példákat. A feladatok megoldásánál a bemenő adatok elemzése, az algoritmus megépítése kiemelt fontosságú, valamint a függvények bemenő és visszaadott értékei közötti kapcsolatok: tehát, hogy egy belső függvény visszaadott értéke egy hozzá külső függvény argumentuma, míg végül a legkülső függvény visszaadott értéke a képlet outputja, az az érték, ami kiíratásra kerül.

A táblázatkezelő programok népszerű szlogenjével szemben, mely szerint az újabb és újabb függvényektől lesz jobb a program, a Sprego programozás a nyelv egyszerűségére helyezi a hangsúlyt. Azt mutatja meg, hogy a minimális eszközigényt összepárosítva az algoritmusok építésével és az eredmények tesztelésével, egy valós problémamegoldásra alkalmas eszközt teremthetünk. Mérési eredményeink bizonyítják [6], [10], [11], [12], hogy ez a mély megközelítésű metakognitív problémamegoldási módszer lényegesen hatékonyabb, mint a felületi megközelítések, csökkentve ezzel a hiba érzékeny dokumentumok számát, a létrehozáshoz és értelmezéshez szükséges időt, emberi és gépi erőforrás igényt.

## Irodalom


[1] *Az olasz, magyar, görög után ezúttal egy holland tanulmány a tudatlanság áráról*. Az ECDL Alapítvány március 9-i sajtóközleménye alapján. Mi újság. 2012. április. http://njszt.hu/sites/default/files/mi_ujsag_2012_aprilis.pdf. (2013.08.30.)

[2] Ben-Ari, M.: *Bricolage Forever!* PPIG 1999. 11th Annual Workshop. 5–7 January 1999. Computer-Based Learning Unit, University of Leeds, UK. http://www.ppig.org/papers/11th-benari.pdf. (1999)

[3] Ben-Ari, M. & Yeshno, T.: *Conceptual models of software artifacts*. Interacting with Computers 18, pp. 1336–1350. (2006)

[4] Biró, P. & Csernoch, M.: *Deep and surface structural metacognitive abilities of the first year students of Informatics.* 4th IEEE International Conference on Cognitive Infocommunications, Proceedings, Budapest, 521–526. (2013a)

[5] Biró, P. & Csernoch, M.: *Elsőéves informatikushallgatók algoritmizáló készségei*. XXIII. Nemzetközi Számítástechnika és Oktatás Konferencia - SzámOkt 2013, EMT, 154–159. . (2013b)

[6] Biró, P. & Csernoch, M.: Táblázatkezelés algoritmikus megközelítése. Kiss Árpád Emlékkonferencia Tanulmánykötete 2013, Debrecen. (2014).

[7] Booth, S.: Learning to program: A phenomenographic perspective. Gothenburg, Sweden: Acta Universitatis Gothoburgensis. (1992)

[8] Csernoch, M.: Programozás táblázatkezelő függvényekkel. Sprego. Műszaki Könyvkiadó, Budapest. (2014)

[9] Csernoch, M. & Balogh, L.: Algoritmusok és táblázatkezelés. Tehetséggondozás a közoktatásban az informatika terén. Magyar Tehetségsegítő Szervezetek Szövetsége, Budapest. ISSN 2062-5936. (2011)

[10] Csernoch, M. & Biró, P.: *Button-up technikák hatékonyságának vizsgálata informatika szakos hallgatók táblázatkezelés-oktatásában*. Szerk: Kozma Tamás és Perjés István, Új kutatások a neveléstudományokban 2012, ELTE Eötvös Kiadó, pp. 369–392. (2013a)

[11] Csernoch, M. és Biró, P.: *Teachers' Assessment and Students' Self-Assessment on The Students' Spreadsheet Knowledge*. EDULEARN13 Proceedings July 1st-3rd, 2013 — Barcelona, Spain. Publisher: IATED, pp. 949–956. (2013b)

[12] Csernoch, M. & Biró, P.: *Spreadsheet misconceptions, spreadsheet errors*. Oktatáskutatás határon innen and túl. HERA Évkönyvek I., ed. Juhász Erika, Kozma Tamás, Publisher: Belvedere Meridionale, Szeged, 2014, pp. 370–395. (2014)

[13] Csernoch, M. & Biró, P. 2014. Digital Competency and Digital Literacy is at Stake, ECER 2014 Conference, 1–5. September, 2014, Porto, Portugal.

[14] *Informatics education: Europe cannot afford to miss the boat*. Report of the joint Informatics Europe & ACM Europe Working Group on Informatics Education April 2013. http://germany.acm.org/upload/pdf/ACMandIEreport.pdf. (2014.02.02.)

[15] *IEEE&ACM Report 2013. Computer Science Curricula 2013*. (2013) The Joint Task Force on Computing Curricula Association for Computing Machinery (ACM) IEEE Computer Society. Letöltés: http://www.acm.org/education/CS2013-final-report.pdf. (2014.04.04.)

[16] Message, R.: *Programming for humans: a new paradighm for domain-specific languages*. Technical Report. UCAM-CL-TR-843. ISSN 1476-2986. University of Cambridge, Computer Laboratory. (2013)

[17] Panko, R. R.: *What We Know About Spreadsheet Errors*. Journal of End User Computing's. Special issue on Scaling Up End User Development. (10)2, pp. 15–21. (2008)

[18] Panko, R. & Aurigemma, S.: *Revising the Panko-Halverson taxonomy of spreadsheet errors*. Decis. Support Syst. 49, 2 (2010), pp. 235–244. (2010)

[19] Powell, S. G., Baker, K. R. & Lawson, B.: *A critical review of the literature on spreadsheet errors*. Decision Support Systems, 46(1), pp. 128–138. (2008)

[20] Powell, S. G., Baker, K. R. & Lawson, B.: *Errors in operational spreadsheets*. Journal of Organizational and End-User Computing, 1(3), pp. 4–36. (2009a)

[21] Powell, S. G., Baker, K. R. & Lawson, B.: *Impact of errors in operational spreadsheets*. Decision Support Systems, 47(2), pp. 126–132. (2009b)

[22] *Report of JPMorgan Chase & Co. Management Task Force*. Regarding 2012. CIO Losses. http://files.shareholder.com/downloads/ONE/2272984969x0x628656/4cb574a0-0bf5-4728-9582-625e4519b5ab/Task_Force_Report.pdf. (2014.05.17.)

[23] Sestoft, P.: Spreadsheet technology. Version 0.12 of 2012-01-31. IT University. (2011)

[24] Tort, F.: *Teaching Spreadsheets: Curriculum Design Principles*. In S. Thorne (Ed.), lProceedings of the EuSpRIG 2010 conference: Practical steps to protect organisations from out-of-control spreadsheets, pp. 99–110. (2010)

[25] Tort, F., Blondel, F.-M. & Bruillard É.: *Spreadsheet Knowledge and Skills of French Secondary School Students*. R.T. Mittermeir and M.M. Sysło (Eds.): ISSEP 2008, LNCS 5090, pp. 305–316, 2008. Springer-Verlag Berlin Heidelberg. (2008)

[26] Van Deursen, A. & Van Dijk, J.: CTRL ALT DELETE. Lost productivity due to IT problems and inadequate computer skills in the workplace. Enschede: Universiteit Twente.

http://www.ecdl.ch/fileadmin/ECDL/CH/Dokumente/Studie_CTRL-ALT-DELETE-en.pdf. (2014. 05. 18.)

[27] Wakeling, D.: Spreadsheet functional programming. JFP 17(1), pp. 131–143, 2007. Cambridge University Press. (2007)

[28] Walkenbach, J. & Wilcox, C.: *Putting basic array formulas to work*. http://office.microsoft.com/en-us/excel-help/putting-basic-array-formulas-to-work-HA001087292. aspx?CTT=5&origin=HA001087290. (2012.05.08.)

[29] Walkenbach, J.: Excel2003 Formulas. John Wiley & Sons. (2003)

[30] Warren, P.: *Learning to program: spreadsheets, scripting and HCI*. in Proceedings of the Sixth Australasian Conference on Computing Education – vol. 30, Darlinghurst, Australia. pp. 327–333. (2004)

[31] Wilcox, C. & Walkenbach, J.: *Introducing array formulas in Excel*. http://office.microsoft.com/en-us/excel-help/introducing-array-formulas-in-excel-HA001087290. aspx. (2013.01.18.)

[32] Wing, J. M.: *Computational Thinking*. March 2006/Vol. 49, No. 3 Communications of the ACM. (2006)