SZÁMÍTÓGÉPES GONDOLKODÁS FEJLESZTÉSE NEM-TRADICIONÁLIS PROGRAMOZÁSI KÖRNYEZETEKBEN

Egyetemi doktori (PhD) értekezés

A szerző neve: Csapó Gábor

A témavezető neve: Dr. Csernoch Mária

DEBRECENI EGYETEM

Természettudományi és Informatikai Doktori Tanács
Informatikai Tudományok Doktori Iskola

Debrecen, 2019

SZÁMÍTÓGÉPES GONDOLKODÁS FEJLESZTÉSE NEM-TRADICIONÁLIS PROGRAMOZÁSI KÖRNYEZETEKBEN

Egyetemi doktori (PhD) értekezés

A szerző neve: Csapó Gábor

A témavezető neve: Dr. Csernoch Mária

DEBRECENI EGYETEM

Természettudományi és Informatikai Doktori Tanács
Informatikai Tudományok Doktori Iskola

Debrecen, 2019

Ezen értekezést a Debreceni Egyetem Természettudományi Doktori Tanács Informatikai Tudományok Doktori Iskola Alkalmazott információ technológia és elméleti háttere programja keretében készítettem a Debreceni Egyetem természettudományi doktori (PhD) fokozatának elnyerése céljából.
Nyilatkozom arról, hogy a tézisekben leírt eredmények nem képezik más PhD disszertáció részét.

Debrecen, 2019.12.20.

Csapó Gábor

Tanúsítom, hogy Csapó Gábor doktorjelölt 2016 - 2019 között a fent megnevezett Doktori Iskola Alkalmazott információ technológia és elméleti háttere programjának keretében irányításommal végezte munkáját. Az értekezésben foglalt eredményekhez a jelölt önálló alkotó tevékenységével meghatározóan hozzájárult. Nyilatkozom továbbá arról, hogy a tézisekben leírt eredmények nem képezik más PhD disszertáció részét.
Az értekezés elfogadását javaslom.

Debrecen, 2019.12.20.

Dr. Csernoch Mária

Számítógépes gondolkodás fejlesztése nem-tradicionális programozási környezetekben

Értekezés a doktori (PhD) fokozat megszerzése érdekében
az Informatika didaktika tudományágban

Írta: Csapó Gábor okleveles informatikatanár - könyvtárpedagógia-tanár

Készült a Debreceni Egyetem Informatikai Tudományok doktori iskolája (Alkalmazott információ technológia és elméleti háttere programja) keretében

Témavezető: Dr. Csernoch Mária

A doktori szigorlati bizottság:

elnök:	Dr.	.....
tagok:	Dr.	.....
	Dr.	.....

Az értekezés bírálói:

	Dr.	.....
	Dr.	.....

A bírálóbizottság:

elnök:	Dr.	.....
tagok:	Dr.	.....
	Dr.	.....
	Dr.	.....
	Dr.	.....

Az értekezés védésének időpontja: .....

Tartalomjegyzék

1. Bevezetés

Az informatika szerves részét képezi a mindennapi életünknek. Behálózza a magánélettől kezdve a munkavégzésen át valamennyi aspektusát mindennapjainknak. Ennek megfelelően az informatikával való kapcsolatunk kiemelt fontossággal bír, akárcsak azok a készségek, amelyek kulcsfontosságú szerepet játszanak az informatika világában: az algoritmikus készség és a számítógépes gondolkodás (Wing, 2006a, 2006b). Az információs társadalomban való élet megköveteli, hogy az emberek információkat rögzítsenek, kezeljenek és megjelenítsenek. Ezen folyamatokat leggyakrabban irodai szoftvercsomagok segítségével végzik a felhasználók, melyek több alkalmazásból tevődnek össze, hogy lefedjék az információfeldolgozás során felmerülő valamennyi területet. Bár ezek a szoftvercsomagok folyamatosan bővülnek újabb és újabb programokkal, kijelenthetjük, hogy első megjelenésüktől kezdődően a legfontosabb alkalmazásaik azokra a területekre fókuszálnak, amelyek környezeti tényezőktől függetlenül valamennyi ember életében jelen vannak: a szövegés a táblázatkezelésre. Ezek az alkalmazások kényelmes, modern felülettel rendelkeznek és valamennyi gyakran előforduló problémára kézre eső megoldásokat kínálnak. Sajnos azonban az intuitív felhasználó felület és az előre elkészített, gyors munkavégzést lehetővé tevő funkciók jelenléte és használata nem jelenti azt, hogy a felhasználók tudják is, hogy mit csinálnak, hogy hogyan és miként kell jól szerkesztett, formázott és hibamentes dokumentumokat előállítani. A hibás dokumentumok (részletesen tárgyalva a 3.3 fejezetben) számos veszélyt hordoznak magukban, melyek közül az anyagi és emberi erőforrások indokolatlan pazarlásra a leginkább kézenfekvő. Felületesen vizsgálva ezek a szoftverek egyszerűnek és használatuk könnyen elsajátíthatónak tűnik. Azonban értő használatuk mögött algoritmusok húzódnak meg, melyek megtervezéséhez és végrehajtásához minden felhasználónak szüksége

van számítógépes gondolkodásra és algoritmikus készségre. Ezen készségek kialakítása elsősorban az informatikaoktatás feladata. A Nemzeti Alaptantervben (OFI, 2012) és a Kerettantervekben (OFI, 2013) megfogalmazott követelményrendszerekre építve, ciklikusan visszatérő témakörökben szerepel ezen tematikus egységek oktatása.

Az algoritmusok létrehozását leggyakrabban a programozás összetett folyamatrendszerével hozzuk kapcsolatba (Aleksić & Ivanović, 2016; Hromkovič, 2009; Soloway, 1993), amely szintén szerves részét képezi az informatikaoktatásnak. Napjainkban megfigyelhető egy, már korábban megjelent irány újbóli elterjedése: a vizuális programnyelvek használata (3.5.3 fejezet). Ezek a nyelvek grafikus elemekkel reprezentált alternatívákat kínálnak a hagyományos szöveg alapú programozási módszerekkel szemben és egyre gyakrabban találkozhatunk velük összetett, sokszor saját grafikus motorral rendelkező fejlesztői környezetekben.

Egy ideális oktatási környezetben a tanulók algoritmikus készségének és számítógépes gondolkodásának fejlesztése integrálva van az informatikaoktatás valamennyi témakörébe (Szlávi & Zsakó, 2019). Az irodai szoftverek algoritmikus szemlélettel történő oktatása (különös tekintettel a kapcsolódó témakörök arányára a tantervekben) kiemelt potenciált hordoz magában. A gyakorlatban azonban sok oktató elszalasztja a lehetőséget, hogy ezeket az alkalmazásokat erre a területre fókuszálva építse be az oktatási folyamatokba. Ennek számos oka lehet, például mivel úgy gondolják, hogy az érintett készségek kialakítása egyedül a programozásoktatás feladata, vagy mert nem ismernek algoritmikus szemléletű módszertanokat (Panko, 2013).

Korábbi kutatások azt bizonyítják (Biró & Csernoch, 2013; Biró et al.,2015; Csernoch et al., 2015), hogy a tradicionális módszerekre támaszkodva a középfokú oktatást elvégző tanulók nem rendelkeznek olyan szintű algoritmikus készséggel és számítógépes gondolkodással, amelyre hatékonyan tudnának támaszkodni a későbbi tanulmányaik, vagy a mindennapi életben felmerülő problémáik megoldása során. Ez sokszor olyan következményekkel jár, mint a magas lemorzsolódási arány vagy a középiskolai tananyag újbóli áttekintésének szükségessége a felsőoktatásban. Infor-

matika tanárként magam is fontosnak tartom a korábban említett két készség kialakítását és fejlesztését a tanulókban. A kutatásaink, valamint oktatási szakmai gyakorlatom rávilágított arra, hogy az említett problémák megoldását az informatikaoktatás során alkalmazott megközelítésekben és módszerekben kell keresni.

1.1. Célkitűzés

Munkámmal a célom, hogy olyan módszertanokat vezessek be és vizsgáljak az informatikaoktatás folyamatába, amelyek hatékonyabban képesek fejleszteni a tanulók algoritmikus készségét és számítógépes gondolkodását, mint a jelenleg elterjedt hagyományos megközelítések. Kutatásaimmal célom a szövegkezelés, a táblázatkezelés, valamint a programozás témakörök oktatására kifejlesztett alternatív módszerek megismerése, megismertetése, vizsgálata, hatékonyságuk elemzése, illetve a munkafolyamatok során szerzett tapasztalatok és eredmények fényében azok továbbfejlesztése. Ezek mellett kiemelkedően fontosnak tartom munkámmal kihangsúlyozni, hogy az algoritmikus készség és a számítógépes gondolkodás fejleszthető alkalmazói környezetekben, valamint akár számítógép használata nélkül is.

2. Tézisek

[T1] A Spreadsheet Lego módszertan szignifikánsan hatékonyabb a táblázatkezelés oktatására, mint a hagyományos, felületorientált megközelítések.

[T2] A séma-alapú algoritmusépítésre alapozott táblázatkezelői módszertan oktatása hosszú távú tudást alakít ki.

[T3] Az első éves informatika szakos hallgatók nem tudnak támaszkodni a közoktatásban elsajátított szövegkezelői ismereteikre.

[T4] Az esemény-utasítás alapú vizuális programozás oktatása hatékonyabban fejleszti a tanulók algoritmikus készségét, mint a blokk alapú oktatási célú programnyelvek alkalmazása.

3. Elméleti megalapozás

Az informatikaoktatás egyik központi kérdésköre a tanulók felkészítése a modern társadalomban megjelenő informatikai problémák megoldására, valamint a tanulók algoritmikus készségének és számítógépes gondolkodásának fejlesztése. A középiskola 10. évfolyamában véget ér a közismereti informatika oktatása (OFI, 2013) és a középszintű oktatás hátralévő két tanévében csak fakultáció keretében tanulható tovább a tantárgy. Azokról a tanulókról, akik ezt a fakultációt választják és teljesítik a közép- vagy emelt szintű informatika érettségi követelményeit feltételezhető, hogy rendelkeznek azokkal a készségekkel, amelyekre támaszkodhatnak a számítógépes problémamegoldás során. Kutatási eredmények igazolják (Biró & Csernoch, 2013; Biró et al., 2015; Csernoch & Biró, 2014a, Csernoch et al., 2015), hogy a gyakorlatban ezek a diákok sokkal magasabbra becsülik a saját tudásukat, mint amilyen ismeretekkel ténylegesen rendelkeznek. Továbbá, az általuk birtokolt informatikai tudás olyan felszínesnek és a kialakított készségek tekintetében megalapozatlannak mutatkozik, amelyre sem a mindennapi életükben, sem a későbbi tanulmányaikban nem tudnak építeni.

A digitális korban történő oktatásban nem hagyhatjuk figyelmen kívül a Pedagogical Content Knowledge (PCK) és Technological Pedagogical Content Knowledge (TPCK) szerkezeteket sem (Angeli, 2013; Shulman, 1986, 1987). Az informatikaoktatásban megjelenő témaköröket figyelembe véve egyetlen tanártól sem várható el, hogy professzionális ismerője és művelője legyen minden érintett területnek. Következetesen, a témakörökhöz kapcsolódó tartalmi elemek kezelése és oktatása azon tanárok feladata lenne, akik az adott elemnek szakértői. Például a szövegszerkesztés témakörben gyakran felmerülő helyesírás ellenőrző funkció értő használatát hatékonyabb lenne, ha az anyanyelvet oktató tanárok oktatnák, akik a helyesírást és a nyelvtani szabályokat is tanítják. Hasonló példaként szolgálhat a grafikus szoftverek oktatása, amelyeket a rajztanárok sokkal mélyebb grafikai és képzőművészeti szakértelemmel tudnának végezni, mint az informatika tanárok.

3.1. Számítógépes gondolkodás

A problémamegoldó megközelítések, tipológiák és a jelenleg alkalmazott hagyományos módszerek tárgyalása előtt szükségesnek tartjuk a számítógépes gondolkodás fogalmának körbejárását. Wing (2006b) a következőképp foglalta össze: „Computational Thinking is the thought processes involved in formulating a problem and expressing its solution in a way that a computer – human or machine –can effectively carry out.”.

A számítógépes gondolkodást más megfogalmazásban, annak szerves részén, a számítógépes problémamegoldáson keresztül is megragadhatjuk: „Computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science. Computational thinking includes a range of mental tools that reflect the breadth of the field of computer science.” (Wing, 2006a). Kiemelnénk, hogy az algoritmikus készség szintén kapcsolódik a számítógépes gondolkodáshoz, mint a számítógépes problémamegoldás része.

Mivel a számítógépes gondolkodás nem csak a mindennapi életben kulcsfontosságú, hanem munkánk során is központi szerepet tölt be, ezért annak bővebb jellemzését és részeit (Wing, 2006a) az általunk vizsgált módszerekkel való kapcsolatán keresztül ismertetjük.

- Koncepció alapú számítógépes problémamegoldásra helyezzük a hangsúlyt, nem kizárólagosan hagyományos programozási környezetekben keressük a megoldási lehetőségeket.
- Alapvető problémamegoldási technikákat tanítunk, sémaépítésre, valamint ezen sémák gyors-gondolkodás alapú előhívására törekszünk, szemben a gépiesen bemagolt frázisok ad-hoc jellegű használatával.
- Az emberi problémamegoldó képességekre, és nem a gépi „logikára” fókuszálunk, úgy, hogy a felépített algoritmusok számítógépen kódolhatók legyenek.
- Alkalmazzuk és kombináljuk a matematikai és a mérnöki gondolkodást, mindezt korlátozva a kiválasztott feladatok hatókörére.

- A problémamegoldásra helyezzük a hangsúlyt és nem az eszközökre, tehát az algoritmus megépítését követően választunk eszközt/eszközöket és keressük ezek közül a leginkább hatékonyat. Tesszük ezt szemben a felületi megközelítésekkel, ahol az eszköz van a fókuszban.
- A számítógépes gondolkodás és ezen készségek fejlesztése nem kizárólag a professzionális informatikusok privilégiuma. Mint alapkészség, erre, az információs társadalomban mindenkinek szüksége van a hatékony számítógépes problémamegoldáshoz (életkortól, szakmai érdeklődéstől, élethelyezetektől függetlenül) és mindenhol (tématerülettől, szakképzettségtől és platformtól függetlenül) alkalmazható.

3.2. Problémamegoldás

Az ACM & IEEE jelentés (2013) három tudásszintet határoz meg:

- Ismeret: A tanuló megérti a koncepciót. Ez a szint alapvető tudatosságot foglal magába a koncepciót illetően, mely azonban még nem biztosítja annak gyakorlati alkalmazását. A következő kérdésre ad választ: „Mit tudunk erről?”
- Használat: A tanuló képes használni vagy alkalmazni egy eszközt konkrét eseteken. A következő kérdésre ad választ: „Mit tudunk megcsinálni?”
- Értékelés: A tanuló képes megvizsgálni egy koncepciót több nézőpontból és indokolni a választását egy konkrét problémamegoldási megközelítést illetően. Ez a szint többet foglal magába, mint egy koncepció használata: a készséget, amely segítségével a tanuló kiválaszt egy alternatív megközelítést általa megértett alternatívák közül. A következő kérdésre ad választ: „Miért csináljuk azt?”

A szintek definíciói párhuzamosságot mutatnak a Pólya (1954) által kidolgozott koncepció alapú problémamegoldó módszerrel. A két megközelítés között azonban némi eltérést fedezhetünk fel. Amíg Pólya négy lépést határozott meg, kiemelve a második lépés (tervezés) fontosságát, addig az ACM & IEEE jelentés egybe foglalja a megértés és tervezés fázisokat (Csernoch & Biró, 2015a).

A számítógépes problémamegoldási megközelítések és gyakorlatok vizsgálata szükségessé teszi azok egy, az informatika didaktika szakterületéhez definiált számítógépes problémamegoldás tipológiához való kapcsolását (Csernoch & Biró, 2015a; Csernoch, 2017).

Mély (DA – Deep Approach) megközelítés:

- Koncepció alapú: A koncepciók felismerése és megértése. A koncepciókon keresztül a probléma megértése, a megoldás megtervezése, a terv végrehajtása és annak ellenőrzése, visszatekintés.
- Számítógépes algoritmikus és debugging alapú (CAAD – Computer Algorithmic and Debugging): Algoritmusok építése, a kimeneti értékek diszkussziója és debugging folyamata, illetve sémák építése, asszociációja és elraktározása.

Felszíni megközelítés (SA – Surface Approach):

- Algoritmikus alapú: A probléma megoldásához szükséges számítási módok előhívása és kivitelezése a probléma megértése nélkül.
- Információ alapú: Az eszközök részleteire fókuszálás, a problémák felismerését vagy megértését mellőzve. A feladatmegoldással feltételezett kapcsolatban álló információk összegyűjtése.
- Próbálkozás és varázsló alapú (TAEW – Trial-And-Error Wizard): Tisztán a szoftverek felületén (gyakran valódi cél nélküli) navigálás. Kapcsolódik a problémamegoldáshoz, de valódi problémamegoldás a legtöbb esetben nem történik. A felületeken történő navigáció kapja a fő hangsúlyt és bármilyen elért eredményt elfogadottnak tekint.

A mély megközelítés koncepció alapú eleme Pólya munkásságára alapoz (Pólya, 1954), továbbá a koncepció alapú és a felszíni megközelítés algoritmikus és információ alapú elemeivel Case & Gunstone problémamegoldó tipológiájában is találkozhatunk (Case & Gunstone, 2002, 2003), valamint hasonló formában Booth funkcionális programozáshoz kapcsolódó tipológiájában is jelen vannak (Booth, 1992).

A számítógépes problémamegoldás tipológiájának megközelítéseit érdemes a mathability fogalmán keresztül is megvizsgálni: „This usage has basically two

forms: in some cases we use existing functions and methods provided by a system, and we apply these tools to solve the problems. Another possibility is, if we, based on existing means of the system, develop new programs and functions for solving new problems." (Baranyi & Gilanyi, 2013). Ez alapján a megközelítések a következő mathability szintekbe sorolhatók be, a legmagasabbtól indulva (Baranyi & Gilanyi, 2013; Biró & Csernoch, 2015a, 2015b; Chmielewska et al., 2016):

– Koncepció alapú (DA) – 5. szint,
– Számítógépes algoritmikus és debugging alapú (DA) – 4. szint,
– Algoritmikus alapú (SA) – 3. szint,
– Információ alapú (SA) – 2. szint,
– Próbálkozás és varázsló alapú (SA) – 1. szint.

A probléma, az input és output adatok közötti kapcsolatok megértéséhez, valamint az algoritmus létrehozásához folyamatos kommunikációra van szükség az 5. és 4. szintek között, melyet a 3. szint bekapcsolódása egészít ki. Az elemző lassú gondolkozás folyamataiba a mély megközelítés elemei tartoznak bele (tervezés, behelyettesítés és diszkusszió), illetve a felszíni megközelítés információ és próbálkozás és varázsló alapú elemei. A rutin gyors gondolkozás előhívására egyedül a felszíni megközelítés algoritmikus szintje szolgál (Csernoch, 2017; Kahneman, 2011).

Jelenleg az informatikaoktatás elsődlegesen az ACM & IEEE jelentés második szintjére fókuszál, ezzel figyelmen kívül hagyva az első szintet és elérhetetlenné téve a harmadikat. A számítógépes problémamegoldás tipológiáját vizsgálva pedig a próbálkozás és varázsló alapú felszíni megközelítés a legelterjedtebb a diákok és a felhasználók körében, ami a legalacsonyabb mathability szinttel rendelkezik. Ebből a gyakorlatból is következik, hogy az informatikaoktatás egyik központi problémája, hogy elsődlegesen a felületi elemek bemutatására és megtanítására koncentrál (Angeli, 2013; Ben-Ari, 1999). Emellett olyan előre elkészített, probléma-specifikus megoldásokat helyez középpontba, amelyek egy adott probléma megoldására alkalmasak, de nem teszik lehetővé a megoldás lépéseinek mélyebb elemzését (Csernoch & Biró, 2014a, 2018; Csernoch, 2017; Dancsó & Korom, 2013; ECDL Foundation,

2016; ICAEW, 2016; JobTestPrep, 2016; Katz, 2010; Microsoft, 2016, 2019; Test ECDL, 2019; Walkenbach, 2010). Ez a gyakorlat nem alakít ki hosszú távú tudást a tanulókban (Carr, 2011; National Research Council, 2000), valamint nem teszi lehetővé a használatát a rutin gyors- és elemző lassú gondolkodásnak (Kahneman, 2011). Az így szerzett, kapcsolódási pontok nélküli ismeretek nem alkalmasak új kontextusban való alkalmazásra, más problémák megoldására és eltérő felületek kezelésére akár iskolai környezetben, akár azon kívül értelmezve. Amennyiben a környezet megváltozik, vagy a tanulóknak a tudásukat egy valós élethelyzetben felmerülő probléma megoldására kellene alkalmazniuk, képtelenek azt megtenni (Csapó, 2003). Ezt a folyamatot erősíti a tankönyvek felépítése és szakmai tartalma is (Anderson, 2017; Dancsó & Korom, 2013; Garrett, 2015). A tanulók készen kapják a megoldásokat, melyeknek csak a felületeken elfoglalt helyét kell megjegyezniük és

ezzel a gondolkodás szükségessége a minimálisra csökken. A tanulók így képtelenek lesznek egy tudatosan megtervezett algoritmus létrehozására a megoldáshoz (Champagne et al., 1983; Panko, 2013). További problémaként merül fel a „sunk-cost fallacy" effektus jelenléte (Kahneman, 2011): A tanulók, tanárok és felhasználók a felületi megközelítések elsajátításába korábban befektetett idővel és energiával indokolják az alternatív módszerek iránt tanúsított zártságukat (Chen et al., 2015). Emellett a Dunning-Kruger hatás is megfigyelhető: a saját tudásukat annál hajlamosabbak túlbecsülni a tanulók és a tanárok, minél kevesebb tényleges tudással rendelkeznek az érintett témakörökről (Kruger & Dunning, 1999). Ez odáig vezethet, hogy a tanulókban kialakulhat egy kényelmes, változásra nem motivált állapot a nem tudásukkal kapcsolatban.

3.3. Hibás dokumentumok

Az előző fejezetben ismertetett fókusz az informatikaoktatásban azt a gyakorlatot szorgalmazza, hogy a tanulók és a felhasználók dokumentumokat „barkácsoljanak" (Ben-Ari, 1999; Csernoch, 2009, 2010, 2011, 2017; Csernoch & Biró, 2016; Panko, 2016; Panko & Port, 2013), más szavakkal olyan hibás dokumentumokat hozzanak létre, amelyek gyakran nélkülöznek minden előzetesen megtervezett és kialakított

dokumentumszerkezetet és melyek elsődleges szerepe az adatok megjelenítése. A fókusz az adatok helyes kezeléséről áthelyeződik azok (látszólag) helyes prezentálására és formázására. Az így kialakított dokumentumok módosításkor, vagy műveletvégzéskor szétesnek, hibákat jelenítenek meg és utómunkák hosszú sorát teszik szükségessé, amellett, hogy nem hozzáértő benyomást keltenek alkotójukról. Gyakran előforduló hiba szöveges dokumentumok előállításakor a szerkezeti elemek hibás használata (Ben-Ari, 1999; Ben-Ari & Yeshno, 2006; Bujdosó & Csernoch, 2014; Csernoch, 2009, 2010, 2011; Csernoch & Bujdosó, 2009): üres bekezdések térköz helyett, középre igazítás szóközzel, hibás tabulátorhasználat, behúzás használata margóbeállítás helyett stb. (Függelék 15.1). További hibaforrásként merülnek fel a különféle nyelvspecifikus, nyelvtani sajátosságok melyek összetettebbek, mint a mesterséges nyelvek szabályai. Ezen utóbbi ismeretek megtanítása nem az informatikaoktatás feladata, azonban az általa okozott hibák megjelenését és szükséges kijavítását informatikai környezetben nem kerülhetjük el. Táblázatok kialakításakor a nem megfelelő adattípus használatával, a nem normalizált adattáblák kialakításával és a probléma-specifikus táblázatkezelői függvények nem hozzáértő alkalmazásával is gyakran találkozhatunk (Függelék 15.2). Egyes esetekben a hibás dokumentumok a módosításukhoz szükséges időigényen túlhaladva, anyagi károkat is okozhatnak az egyénnek vagy egy szervezetnek, akár csődbe is juttatva azt (Csernoch & Biró, 2015b; EuSpRIG, 2019).

3.4. Unplugged informatika

Hagyományos és logikusan következtethető megközelítés, hogy az informatikai készségek kialakításának elsődleges terepe a számítógépek előtt végzett munka. Ez a gyakorlat azonban figyelmen kívül hagy számos érzékszervet, amelyek bevonása az oktatási folyamatba potenciális motivációs és hatékonyságnövelő hatást hordoz magában (Kátai et al., 2014; Shams & Seitz, 2008). Korábbi próbálkozások igazolják, hogy nem újkeletű elgondolás az informatikaoktatás folyamatát olyan eszközök használatával támogatni, amelyek kiegészítik vagy helyettesítik a számítógépes

munkavégzést (Bell & Newton, 2013; Biró & Csernoch, 2017a, 2017b). Ezen eszközök használatának a fő szerepe a tanulók gondolkodásának kialakításában, formálásában és stimulálásában van. A szoftverkörnyezet leválasztásával lehetővé teszik, hogy a feladat elemzése és megoldása során kizárólag a folyamat mögött meghúzódó algoritmusokra fókuszáljanak. A felhasználói felület, a beépített (sokszor speciális célú) funkciók az algoritmizálás szempontjából legtöbbször zavaró tényezőkként vannak jelen.

Az unplugged eszközök számos formát ölthetnek (Függelék 15.3 és 15.4), sokszínűségük meglehetősen széles skálán mozog. Fontos hangsúlyozni, hogy az unplugged eszközök megalkotásának és integrálásának folyamatában a fókuszt elsődlegesen nem az eszközök összetettségére és bonyolultságára kell helyezni, hanem sokkal inkább azok módszertani beépítésére a feladatmegoldásba. Az általunk alkalmazott unplugged eszközökkel történő feladatmegoldás menetét az adott témakörhöz kapcsolódó módszertan bemutatásánál részletezzük (3.5.1.1 és 3.5.2.2 fejezetek).

3.5. Módszertani eszközök

Az informatika oktatásában jelenleg széleskörűen alkalmazott felületorientált megközelítések nem fejlesztik a tanulók algoritmikus készségét és számítógépes gondolkodását olyan mértékben, amely elvárt a modern információs társadalomban akár a munkaerőpiacon, akár a továbbtanulást tekintve (Biró & Csernoch, 2013; Biró et al., 2015; Csernoch et al., 2015).

Az alkalmazói ismeretek oktatása során használt irodai szoftvercsomagok összetett és komplex alkalmazásokat foglalnak magukba, melyek képességeinek teljeskörű megismerése és kihasználása sokszor a professzionális felhasználók számára is nehéz feladatnak bizonyul. Az érintett témaköröket a tanulók leggyakrabban a Microsoft Office (Microsoft, 2019b), a LibreOffice (The Document Foundation, 2019), valamint a kollaborációs munkát könnyen lehetővé tevő Google Dokumentumok (Google, 2019) szoftvercsomagok használatán keresztül tanulják. A programozásoktatás az alkalmazói ismeretekhez képest szélesebb eszköztárral rendelkezik, melybe

a hagyományos programfejlesztői környezeteken és programnyelveken túl az oktatási céllal készült nyelvek és környezetek (Educational Programming Language – EPL) is beletartoznak (3.5.3 fejezet).

Az érintett szoftverek összetettségéből és gazdag lehetőségeikből adódóan következik, hogy figyelmünket elsődlegesen az ezen eszközökre támaszkodó módszerekre és módszertanokra kell fókuszálni a tanulók alacsony teljesítményeinek javítása érdekében. A tárgyalt problémák indokolttá teszik az informatikaoktatásban jelenleg alkalmazott módszerek újragondolását és alternatív megközelítések keresését.

3.5.1. Spreadsheet Lego

A Spreadsheet Lego (röviden: Sprego) egy sémaközpontú, magas mathability (Baranyi & Gilanyi, 2013; Biró & Csernoch; 2015a, 2015b; Chmielewska et al., 2016) módszertan, amely az algoritmikus szemléletet előtérbe helyezve oktatja a táblázatkezelés témakört (Csernoch, 2014). A Sprego a táblázatkezelői problémák megoldása során Pólya (1954) négylépéses problémamegoldó stratégiájára támaszkodik:

1. A probléma megértése: Az adatok és a feladat elemzése. A feladat követelményei alapján elérni kívánt output megfogalmazása.
2. A terv megépítése: A megoldáshoz szükséges algoritmus megtervezése. Az egyes algoritmus lépések közötti input és output értékek hangsúlyozásával.
3. A terv megvalósítása: Az algoritmus kódolása táblázatkezelői környezetben, függvények használatával.
4. Visszatekintés: A kapott eredmények értékelése, elemzése, tesztelése, diszkusszió.

A módszer alapelve, hogy a számos probléma-specifikus táblázatkezelői függvény megtanulása és alkalmazása helyett általános célú függvények kis csoportjára támaszkodva építik fel a tanulók a problémák megoldásait. Ezeket a függvényeket három kategóriába soroltuk felhasználási területük és funkcióik alapján (1. táblázat).

1. táblázat: A Sprego módszertanban alkalmazott függvények három kategóriába sorolva.

Sprego szöveg	Sprego szám	Sprego pro
BAL()	MIN()	HA()
JOBB()	MAX()	INDEX()
HOSSZ()	SZUM()	HOL.VAN()
SZÖVEG.KERES()	ÁTLAG()	HIBÁS()

A Sprego módszertanban alkalmazott 12 függvény elegendő a Kerettantervekben (OFI, 2013) meghatározott témakörspecifikus követelmények teljesítésére. A feladatmegoldás komplexitásától függően a használt függvények halmaza igény szerint tovább bővíthető (2. táblázat), mellyel további programozási koncepciók oktatása is bevezethető a témakörbe.

2. táblázat: További általános célú táblázatkezelői függvények a Sprego módszertan eszköztárának bővítésére.

KICSI()	ÉS()	TRANSZPONÁLÁS()
NAGY()	VAGY()	KEREKÍTÉS()
SOR()	NEM()	VÉL()
OSZLOP()	ELTOLÁS()	INT()

Az általános célú függvényeket egymásba ágyazva összetett képletek, valamint tömbképletek formájában történik az algoritmusépítés (Biró & Csernoch, 2015b, 2017a, 2017b; Csernoch, 2014; Csernoch & Biró, 2014b, 2017; Csernoch et al., 2014a, 2014b; Walkenbach & Wilcox, 2003; Wilcox & Walkenbach, 2003). Kiemelnénk, hogy a módszertan a táblázatkezelői környezetek funkcionális nyelvét alkalmazza (Sestoft, 2011) és nem érinti a makróprogramozás területét. A Sprego kiemelt figyelmet fordít az algoritmusok lépésről-lépésre történő áttekintésére a folyamat során kezelt input és output adatok elemzésével. A feladatmegoldás közben elemi programozási tételekkel dolgoznak a tanulók, melyek számos, tudástranszferben megjelenő egységet magukba foglalnak (Csernoch, 2019a, 2019b). Ezeket az elveket követve a módszertan sémákat alakít ki a tanulókban (Csernoch, 2016; Hermans, 2019; Merriënboer & Sweller, 2005; Skemp, 1971). Ezekre a sémákra támaszkodva más kontextusban, hasonló megoldási algoritmust igénylő feladatokban a tanulók képe-

sek előhívni és alkalmazni korábbi ismereteiket, adaptálva azokat az aktuális probléma megoldására. Ezt segíti elő a módszertan azon gyakorlata, hogy ismétlődő, hasonló problémákat tár a tanulók elé azokat más kontextusban megjelenítve, mint amilyenben korábban már megoldották azokat (Skemp, 1971).

Korábbi, késleltetett utótesztekre alapuló kutatás igazolja, hogy azok a tanulók, akik a Sprego módszertannal tanulták a témakört egy évvel később is 40–50%-os eredményekkel teljesítettek táblázatkezelői feladatokat. Ezzel szemben a hagyományos, probléma-specifikus megközelítéssel tanuló diákok teljesítménye 5–10% volt (Csernoch & Biró, 2015a).

3.5.1.1. Sprego unplugged eszközök

Az algoritmusok építésének megértésének elősegítésére a módszertannal történő munkát unplugged eszközök támogatják (Biró & Csernoch, 2017a). A tanulók elsődlegesen 3D nyomtatással készült matrjoska babákat alkalmaznak. Ezen babák előnye, hogy felépítésükből adódóan kiemelten alkalmasak a témakörspecifikus tudásátadás támogatására és a tanulók bevonására az oktatási folyamatba taktilis módon. Hátrányuk azonban az előállításuk körülményessége, költsége és időigénye, amely számos oktatási intézmény számára problémákat jelent. Ezzel szemben a színes papírlapokból hajtogatott origami csónakok praktikusabb és könnyebben bevezethető eszközöknek bizonyulnak. Fontos kiemelni, hogy az eszközcsoportokon belül minden tárgy egyértelműen elkülöníthető színnel és mérettel rendelkezik, valamint, hogy darabszámuk megegyezik az aktuális algoritmus lépéseinek számával. További lehetőségekként szerepelnek a módszertan eszközkészletében a játékboltban kapható műanyag hordókészletek (hasonló felépítéssel, mint a 3D nyomtatott babák), illetve a fényvisszaverő mellények.

A készletek (baba, origami csónakok, hordók) fő jellemzője azok egymásba ágyazhatósága. A tanulók a feladatmegoldás során tanári instrukciót követve a feladat algoritmusának egy lépését felírják ragasztószalagra (vagy hajtogatott csónakok esetében azok oldalára), majd ráragasztják a megfelelő méretű eszközre. A lépés szö-

vegének feltüntetése mellett fontos szerepe van az adott lépés input és output értékeinek szerepeltetésének is az eszközön. A következő lépés végrehajtása során a módszertan az előző lépésekre támaszkodik. Ennek megfelelően a korábbi eszköz beillesztésre kerül az eszközkészlet egy nagyobb darabjába. Ezzel szimbolizálva az egyes lépések közötti értékátadást (hogyan szolgálnak az alacsonyabb szintű outputok magasabb szintű inputokként), az összetett táblázatkezelői képletek felépítését, valamint a korábbi lépésekre való támaszkodást és a több lépcsőből álló feladatmegoldás menetét. A tanulók az unplugged eszközökkel felépített algoritmus adott lépését elvégzik táblázatkezelői környezetben is, párhuzamosan használva az eszközöket. A munkavégzés után a tanulók szétszedik az eszközkészletet és a rajtuk szereplő felcímkézett ragasztószalagokat (illetve csónakok esetén magukat a csónakokat) a füzetükbe ragasztják. Ezzel a gyakorlattal a módszertan biztosítja a tanulók egységes és megbízható jegyzetelését is.

A mellények segítségével a diákok testmozgással vesznek részt az egyes lépések kidolgozásában, mely folyamatban a mellények a tanulókat – mint adatokat – jelölik meg a feladat specifikációi szerint. A mellények alkalmazása (szemben az egymásba ágyazható eszközökkel) kis csoportokban ideális és a fiatalabb korcsoportokat célozza meg.

3.5.1.2. Sprego semi-unplugged: oktatószoftverek

A módszertan eszközkészletét tovább bővítik a semi-unplugged eszközök (Biró & Csernoch, 2017b). Ide tartoznak a leggyakoribb Sprego táblázatkezelési problémák megoldásait lépésről lépésre bemutató algoritmusvizualizációs eszközök (Csapó & Sebestyén, 2017; Gulácsi & Dienes, 2018; Sebestyén & Csapó, 2018). Korábbi vizsgálatok és kutatások támasztják alá, hogy az oktatószoftverek alkalmazása kizárólag irányított instrukciókkal hatékony (Iacob, 2009; Kecskés, 1987; Wilson et al., 1996). Ez az irányvonal befolyásolta a Sprego semi-unplugged eszközök kialakítását is.

Munkánk során fejlesztettünk egy oktatószoftvert a Sprego módszertanhoz (Csapó & Sebestyén, 2019) azzal a céllal, hogy elősegítsük az összetett táblázatkezelői képletek felépítésének és működésének megértését. Olyan megközelítést választottunk, amelyben valós életből vett példákra alapozott animált grafikai prezentációkon keresztül történik az algoritmusok egyes lépéseinek bemutatása (1. ábra). Követve az unplugged eszközöknél megfigyelhető tendenciát, szoftverünkkel is a többszenzoros ingerlésre törekedtünk. Ezt szolgálja a színekben gazdag grafikai felület és a program egyes részeinél alkalmazott hangeffektek. A fejlesztési folyamat kezdeti fázisaitól kezdve figyelembe vettük a tanári és tanulói visszajelzéseket, melyek kulcsfontosságú szereppel bírtak a felhasználói felület és a tartalomstrukturálás kialakításában. A programunk fő szereplői a módszertanban gyakran felmerülő, unplugged eszközök esetén pedig fizikailag is jelen lévő matrjoska babák. Jelenleg a következő táblázatkezelői problémákat tartalmazza a szoftver eltérő kontextusokba ültetve:

– Feltételes megszámlálás egyenlőséget vizsgálva: Hány azonos színű baba van? (1. ábra)
– Feltételes megszámlálás egyenlőtlenséget vizsgálva: Hány baba ülhet fel az óriáskerékre? (Függelék 15.5)
– Lineáris keresés: Melyik házban lakik a baba? (Függelék 15.6)

1. ábra: A Sprego módszertanhoz fejlesztett oktatószoftver feltételes megszámlálást bemutató, egyenlőségen alapuló prezentációja.

Az algoritmusok egyes lépéseinek követésére beépítettünk az alkalmazásba egy, a táblázatkezelői környezetekben is megtalálható képletkiértékelő panelt (1. ábra, jobb oldal). Ezen a felületen a bemutatóban is látható grafikus elemek kis méretben, kiemelő jelöléseket alkalmazva szemléltetik az egyes lépések vizsgálatait, illetve a kapcsolódó adatokat. A tanulók ezt a felületet használva párhuzamosan követhetik az algoritmusok lépéseit az animációkkal. A program jelenleg angol és magyar nyelven elérhető és letölthető Android készülékekre a Google Play Áruházon keresztül (Csapó & Sebestyén, 2019), valamint egy böngészőben futtatható verziója is rendelkezésre áll. A szoftver támogatja és rugalmasan vált érintőképernyő és egér inputok között, amely széles eszközcsoport támogatását teszi lehetővé, beleértve a tantermi interaktív táblákat is.

Az oktatószoftver számos alkalommal bővült első verziójának megjelenése óta. A kezdeti két bemutató elkészítését követte az egyenlőtlenségen alapuló feltételes megszámlálás prezentációja amellett, hogy a korábbi bemutatók is felülvizsgálatra és számos ponton átalakításra kerültek. Az alkalmazás további fejlesztése folyamatos, amely a jövőbeli verziókra értelmezve magába foglalja új bemutatók implementációját, megértést segítő narráció rögzítését és beépítését, platformok szélesebb körének támogatását, valamint a szoftver több nyelvre történő lefordítását is.

A 2D-s oktatóprogramunk felépítését követve elkészült annak 3D környezetbe átültetett változata (Gulácsi & Dienes, 2018). A program alapvető célja és az algoritmusok bemutatói nem változtak, azonban új, 3D megjelenítést kaptak. Implementálásra került egy „interaktív mód” is, melynek segítségével a tanulókat kérdezve, a helyes választ várva és ellenőrizve léptethetők előre az algoritmusok prezentációi. A 3D-s oktatóprogram jelenleg asztali operációs rendszerekre elérhető. A fejlesztési folyamatban későbbi tervek között szerepel annak átültetése mobil eszközökre, illetve (követve a 2D-s alkalmazás fejlesztési irányvonalait) a bemutatók és elérhető nyelvek számának bővítése.

3.5.1.3. Sprego semi-unplugged: virtuális kollaborációs tér

Az egyéni tanulás támogatására létrehoztunk egy online 3D kollaborációs teret (Csapó, 2017a, 2017b). A tér a MaxWhere 3D platformra (korábbi nevén VirCa – Virtual Collaboration Arena) (Galambos et al., 2011; MISTEMS, 2019) alapulva helyezi el az oktatáshoz szükséges tartalmat virtuális képernyők (webtáblák) segítségével 3D környezetben (Berki, 2018; Horváth, 2018a, 2018b; Lampert et al., 2018). A navigáció a tanulók számára már ismert, videojátékokban is alkalmazott módon valósul meg. A kollaboráció a beépített webtáblákon keresztül történik, a többjátékos videojátékokból ismert koncepció, miszerint minden felhasználó mások által látható avatárokkal van jelen a térben, a MaxWhere környezetben nem implementált. A szoftverkörnyezet sajátosságai jó lehetőségeket kínálnak az oktatásban való felhasználásra, melyet korábbi, hasonló irányultságú munkák igazolnak. A Zeleméri templomrom rekonstruálása és prezentálása 3D térben (Gilányi et al., 2015a, 2015b), illetve a Debreceni Egyetem Egyetemi és Nemzeti Könyvtára által 3D térben elérhetővé tett értékes és ritka dokumentumok (Gilányi & Virágos, 2013) projektek jól szemléltetik a MaxWhere lehetőségeit.

A Sprego módszertant magába foglaló térben a Google Dokumentumok (Google, 2019) szolgáltatáson keresztül jelennek meg a témakörspecifikus információk, feladatok és egyéb tartalmak. A tartalmi strukturálás a térben a tanórák során alkalmazott sorrendet követi: szövegkezelő algoritmusok, feltételes műveletvégzés és lineáris keresés. A tanórákon megoldott feladatokat követve a térbe valamennyi szükséges információt beépítettünk az önálló munkavégzéshez és a feladatok gyakorlásához, valamint a 2D-s oktatószoftverünk böngészőben futó verzióját is elérhetővé tettük. A kidolgozás során célunk az volt, hogy olyan feladatlistát állítsunk össze, amelyet, ha a tanuló önállóan képes megoldani, a témakört megtanultnak tekintheti. Az önellenőrzés eszközeként a feladatok mellett (külön dokumentumban, természetes nyelven megfogalmazva) elérhetővé tettük a problémák megoldásainak algoritmusait is. Ezzel a tartalmi strukturálással egy mindent magába foglaló virtuális tanulási

környezetet hoztunk létre, amelyben annak elhagyása nélkül minden szükséges eszköz a tanuló rendelkezésére áll. A Sprego virtuális kollaborációs tér jelenleg korlátozott formában érhető el, annak publikussá tétele további megelőző vizsgálatokat igényel.

3.5.1.4. Sprego autentikus források

A Sprego módszer egyik további fontos eleme az autentikus adatokkal történő feladatmegoldás. Szemben a tankönyvekben és súgókban alkalmazott dekontextualizált tartalmakkal (Angeli, 2013; Csernoch, 2017), a Spregoban a tanulók szigorúan valós adatokkal dolgoznak, melyeket internetes forrásokból készít elő az oktató. Az előkészítést egyedül is elvégezheti a tanár, illetve amennyiben az iskola helyi tantervére épülő tanmenetek megengedik, más témakörbe integráltan a tanulókkal közösen is végrehajthatja. A folyamat során az interneten található táblázatnak kinéző adatok kerülnek konvertálásra, hogy alkalmasak legyenek táblázatkezelői környezetben történő megjelenítésre és munkára. Fontos kiemelni, hogy a feldolgozott adatok mennyiségét tekintve a módszertan reprezentativitásra törekszik, valós életben előfordulható eseteket szimulálva, éppen ezért nem ritka a több száz, vagy több ezer rekordból álló adathalmazzal végzett munka. Az autentikus megközelítés elősegíti, hogy a tanulók számukra motiváló, érdekes és ismerős adatokkal dolgozzanak (Biró & Csernoch, 2017b; Csernoch & Dani, 2017). A módszertan ezen keresztül teljesíti a tanulók azon igényét, hogy közvetlen kapcsolatot alakít ki a tanult ismeretanyag és a valós világ között (Message, 2013). Továbbá, a konvertált és felhasznált adatok azok tematikájától függően tantárgyközi kapcsolatok kialakítására is alkalmasak.

A Sprego módszertan korosztálytól és platformtól függetlenül alkalmazható, kompatibilis a piacon elterjedt táblázatkezelői környezetekkel. A táblázatkezelés oktatásán keresztül az ismertetett megközelítéseket követve alkalmas arra, hogy előkészítse az informatikaoktatásban az adatbáziskezelés és programozás témaköröket (Hubwieser, 2004; Schneider, 2004, 2005; Sestoft, 2011; Szlávi & Zsakó, 2019; Zsakó, 2006; Wakeling, 2007). Az alacsony informatika óraszámból kiindulva (OFI, 2012, 2013) a közoktatásban megfontolandó felvetés lehet az említett témakörök

nem csupán előkészítésére, de oktatására is alkalmazni a Sprego módszertant. Ilyen esetben további előnyként jelenik meg, hogy a tanulói- és munkakörnyezet változatlan marad, így a rendelkezésre álló időt nem csökkenti az új környezet és formális nyelv elsajátítása, valamint hatékonyabb tanuló előrehaladást tesz lehetővé (Kátai, 2016; Osztián et al., 2017).

3.5.2. Error Recognition Model

A szövegszerkesztés oktatását az Error Recognition Model (röviden: ERM) a szövegszerkesztési hibák oldaláról közelíti meg. A módszertan megépítésének előmérésére a TAaAS projekt szolgált alapul (Biró et al., 2015; Csernoch et al., 2015). Az ERM a programozás oktatásában használt debugging módszert követi, melyben a tanulók hibás forráskódok elemzésével és javításával sajátítják el a tudáselemeket (Gould, 1975; Gould & Drongowski, 1974; Jerinic, 2014). Ebből következik, hogy szakít a hagyományos megközelítésekkel és nem a szövegszerkesztő program felületére és funkcióira fókuszál. A módszertan elsődleges célja a jól formázott dokumentumok kialakítása: olyan dokumentumoké, melyek megfelelnek a nyomtatott formájukkal szemben támasztott elvárásoknak és emellett ellenállnak a szövegbevitelnek; azok tartalmi módosítása nem igényel újra tördelést és formázást. Az egyetlen kivétel ez alól a nagymértékű szövegbevitel (Csernoch, 1997, 2009, 2010, 2011, 2019a).

Az elkészített dokumentumok a nyelvtani szabályok mellett számos egyéb kritériumnak is meg kell, hogy feleljenek. A tipográfiai szabályok ebben nagy szerepet kapnak, külön hangsúlyt fektetve az eltérő interfészekre, amelyeken tartalommegjelenítés történik, annak figyelembevételével, hogy miként tehető könnyen olvashatóvá a tartalom. Ezen túlmenően szem előtt kell tartani, hogy a digitális szövegek több feltételnek kell, hogy megfeleljenek mint a kézzel írott szövegek. Ezen vonatkozásban nem kerülhetjük el a tördelési és formázási szabályokat sem (Csernoch, 2019a). Fontos kiemelni, hogy a táblázatkezelés és programozás eszközeivel szemben a természetes nyelvű szövegek elemzésére és javítására nem áll rendelkezésre

automatizált segítséget nyújtó debugger, ami iránymutatást adhatna a dokumentumok helyes elkészítésében. Az egyetlen, hasonló kategóriába sorolható segítséget a beépített nyelvhelyesség ellenőrzők nyújtják a felhasználók számára, amelyek hatékonysága nagymértékben függ a felhasználók képességeitől, valamint a nyelv sajátosságaitól. Mindezeket figyelembe véve indokolt elvárás, hogy a felhasználók elsajátítsák a digitális szöveges dokumentumokban felmerülő hibák felismerését, javítását és elkerülését.

Munkánk során szövegkezelésről beszélünk, mely alatt a digitális szöveg alapú tartalmak létrehozását, módosítását, terjesztését és az ezen dokumentumokból történő információkinyerést értjük. Az ERM módszertan a felsorolt célokat hibás dokumentumok elemzésén és kijavításán keresztül éri el. A következő kompetenciák alakíthatók ki és fejleszthetők a segítségével (Carretero et al., 2017; Shulman, 1986):

- Megérteni, hogy a szövegkezelés sokkal több annál, mint a felhasználói felületen navigálni és önkényesen alkalmazni olyan funkciókat, amelyek központi helyet foglalnak el a menüszalagokon. A szövegkezelés a tartalomalkotás, kialakítás és szerkesztés egy folyamata, melyet előzetes tervezés előz meg az aktuális és várható igényeknek megfelelően.
- Megérteni, hogy a szövegek módosítása hatékonyabb, ha a szöveget a jól formázott szövegek elve alapján építjük fel.
- Megérteni, hogy a kézi szerkesztések, „barkácsolások” befolyásolják a nyomtatott verziót és olyan hibákat eredményeznek, amelyek nyomtatott formában is azonosíthatók.
- Megérteni, hogy ha tisztában vagyunk a különféle típusú hibakategóriák sajátosságaival, az segít elkerülni azokat.
- Képesnek lenni felismerni a hibákat nyomtatott formában.
- Képesnek lenni felismerni a hibákat elektronikus, forrásdokumentum formában.
- Képesnek lenni megnevezni, osztályozni és eldönteni, hogy a szövegszerkesztési folyamat mely szintjén keletkezett a hiba.
- Képesnek lenni kijavítani a hibákat.
- Képesnek lenni eszközöket alkalmazni a szerzői jogi kérdések vizsgálatára.

– Képesnek lenni olyan szövegeket megtalálni, amelyek hibákat tartalmaznak a felállított követelményeknek megfelelően.

3.5.2.1. Hibaanalízis és -javítás

A természetes nyelvi szövegekben előforduló lehetséges hibák számából adódóan azokat kategóriákba soroljuk. A tanulók a feladatmegoldás során hibás dokumentumokkal dolgoznak, melyek hat hibakategóriákba besorolt hibákat tartalmazhatnak (3. táblázat).

3. táblázat: Az ERM módszertanban alkalmazott hibakategóriák.

Mennyiségi hibák	**Minőségi hibák**
Szintaktikai	Tördelési
Szemantikai	Formázási
Tipográfiai	Stílus

A hibák felismerésében, elemzésében és javításában kiemelt szerepet játszanak a nem nyomtatható karakterek. Ezeknek a karaktereknek a megértése, a szerepük definiálása nagy hangsúlyt kap a módszertannal történő munkavégzésben. Ismeretük hiányában nem lehetséges a jól szerkesztett és formázott dokumentumok létrehozása. A hibakategóriák közül a szintaktikai, szemantikai, tipográfiai és egyes formázási hibák nyomtatott formában, míg a tördelési, stílus és a további formázási hibák digitális formában azonosíthatók.

A kerettantervekben meghatározott követelményeket és ismereteket (OFI, 2013) a dokumentumok kijavításán keresztül a módszertan fokozatosan vezeti be és tanítja. A dokumentumokon végzett feladatok összeállításához a programozásoktatásban elterjedt hibakezelés és hibajavítás módszerét alkalmazza az ERM (Lister, 2016). A tanulók a szövegszerkesztő program valamennyi alapvető funkciójával és lehetőségével megismerkednek a fokozatosan egyre bonyolultabb, több és összetettebb hibákat tartalmazó fájlok kijavításán keresztül. Fontos, hogy a hibák javítása során a módszertan kitér az adott hibajelenséggel járó következményekre, megválaszolva a kérdést, hogy „Miért hiba az, amit kijavítunk?”.

3.5.2.2. ERM unplugged eszközök

A szövegkezelés tanításának támogatására a forrásfájlok nyomtatott verzióját kapják kézbe a tanulók unplugged eszközként.

A feladatmegoldás során a tanulók a saját példányukat átnézik, majd kék tollal megjelölik azokat a hibákat, amelyeket felismernek a dokumentumban. Fontos, hogy a tanulók a hiba megjelölése mellett azt is fel kell, hogy tüntessék a lapon, hogy miért jelölték hibásnak az érintett részt. A kék szín használata jelzi, hogy a hibák azonosítása nyomtatásban történt meg.

A következő lépésben a tanulók bevonják a szövegszerkesztő szoftvert is a folyamatba. A forrásfájlt megnyitják, melyből saját példányt mentenek, majd a nyomtatott laphoz való visszatérés előtt a szövegszerkesztő program „Minden látszik” (vagy azzal egyenértékű) funkcióját bekapcsolják. A nyomtatott dokumentum forrását és szerkezetét látva a diákok ismételten átnézik a dokumentumot. Ezúttal piros színnel tesznek, a korábbi jelöléseik mellé kiegészítő jegyzeteket, melyekkel megjelölik a nyomtatásban nem látható hibákat és ismételten megnevezik, hogy mivel indokolják jelölésüket.

A nyomtatott dokumentumok unplugged eszközként való használata és bevonása a hibafelismerés és -elemzés folyamatába egy átlátható jegyzetet biztosít a tanulók számára és segíti a munkafolyamatok nyomon követését a következő lépésben, a dokumentum kijavításában. Továbbá tudatosítja a tanulókban a következő, a témakörben sokszor felmerülő elvet: Attól, hogy a dokumentum nyomtatásban úgy néz ki, ahogyan szeretnénk, az még nem szükségszerűen jól szerkesztett.

3.5.2.3. ERM autentikus források

Hasonlóan a Sprego módszertanhoz, az ERM-en belül is autentikus forrásokkal dolgoznak a tanulók. A tanórák során használt dokumentumok elsődleges forrása az internet, melyeket felhasználók készítettek egy valós helyzetben, saját szövegkezelői tudásukra támaszkodva. A dokumentumok halmazából az oktató feladata olyan forrásfájlok kiválasztása, amelyek kijavítása illeszkedik a tanítási-tanulási folyamat adott szintjére. Az autentikus dokumentumok felhasználása követi a valós életből

vett példákra támaszkodás elvét. Ezek a szövegek hiteles forrásokként szerepelnek a tanulók előtt, melynek egyik célja éreztetni, hogy nem csupán a tanóra kedvéért előkészített adatokkal történik a munkavégzés. Továbbá arra is rávilágít, hogy számos végfelhasználó nem rendelkezik olyan alapvető szövegkezelési ismeretekkel, amelyek segítségével jól szerkesztett dokumentumok állíthatók elő (Függelék 15.1). Kiemelnénk, hogy az értekezés írásának idejében nem tudunk más olyan kutatócsoportról, amely a szövegkezelés oktatására ilyen módon alkalmazna autentikus forrásokat. A külső forrásokból szerzett dokumentumokon végzett munkából következik, hogy a tanóráknak nem része – szintén hasonlóan a Sprego módszertanhoz – a forrásszövegek begépelése.

Az Error Recognition Model kompatibilis valamennyi szövegszerkesztő környezettel és platformmal, amely rendelkezik olyan funkcióval, hogy a nem nyomtatható karaktereket meg tudják jeleníteni a tanulók. Hasonlóan a Sprego módszertanhoz, platformtól és korosztálytól függetlenül alkalmazható.

3.5.3. Vizuális programozás

A számítógépes gondolkodás és az algoritmikus készségek kialakítására különféle programozási környezeteket alkalmaznak az informatikaoktatásban. Jelen vannak a modern objektum-orientált nyelvek (pl.: C++, C#, Python, Java), de előfordulnak olyan procedurális nyelvek is, amelyeket a modern szoftverfejlesztő ipar csupán elvétve alkalmaz (pl.: Pascal). Ezeknek az imperatív programnyelveknek az elsajátítása nem könnyű feladat kezdő programozók számára, melyet nehezítenek a nyelvek szintaktikai és utasításkészleti sajátosságai (Soloway, 1993; Ben-Ari, 2011). A programozásoktatás hatékonyabbá és könnyebben befogadhatóvá tételére számos oktatási programozási nyelv (EPL) került kifejlesztésre és bevezetésre, melyek közül több is elrugaszkodik a hagyományos szöveg alapú programozási formától. Ezeknek a nyelveknek az alkalmazása napjainkban már széleskörűen elfogadott gyakorlat (Fincher et al., 2010; Fowler et al., 2012; Klassner & Anderson, 2003; Papadakis et

al., 2014; Resnick et al., 2009). Azonban ezen törekvések ellenére a közoktatást elvégző tanulók nem rendelkeznek olyan szintű algoritmikus készséggel, amellyel képesek lennének egyszerű feladatok megoldására (Csernoch et al., 2015).

A vizuális programozás során a felhasználók előre definiált grafikus programnyelvi elemeket használnak a programkód felépítésére. Ez a folyamat előtérbe helyezi az algoritmusok építését anélkül, hogy az adott nyelv szintaktikai szabályaira kellene fókuszálni. A módszer elterjedtségét és szakmai elfogadottságát igazolja, hogy napjainkban számos, piacvezető fejlesztői környezetben van lehetőségünk vizuális programozási nyelveket használni a hagyományos programozás mellett (Epic Games, 2019; Hutong Games, 2019; YoYo Games, 2019). Az egyszerű és látványos megjelenítés, valamint a felépítésükből eredő gyorsabb fejlesztési folyamat ígéretes jelöltekké teszik ezeket a nyelveket az oktatásban való alkalmazásra is (Fesakis & Serafeim, 2009; Kim et al., 2012). Fontos azonban kiemelni, hogy a vizuális programnyelvek gyakran korlátozottabbak lehetőségeiket illetően, mint az imperatív programnyelvek, illetve a felhasználható grafikus építőelemek készlete és funkciója mindig nyelvspecifikus.

A vizuális programnyelvek változatos formái nem kompatibilisek egymással: az egyik fejlesztőkörnyezetben készített kód nem fordítható át egy másik környezetbe. Elemezve a piacon jelenlévő népszerűbb vizuális nyelveket, az alábbi négy kategóriát fogalmazhatjuk meg, melyekbe megjelenési formájuk alapján besorolhatók (Csapó, 2019; Csapó & Sebestyén, 2017):

- Viselkedés alapú nyelvek (behavior based),
- Esemény-utasítás alapú nyelvek (event-action based),
- Blokk alapú nyelvek (block based),
- Csomópont alapú nyelvek (node-based).

A programfejlesztés területén túlmenően a vizuális programozás az oktatási folyamatokban is jelen van. Számos EPL és a hozzá kapcsolódó környezetek a vizuális nyelvek felépítését követik. Erre szolgálnak példaként az alábbi, gyakorlatban is alkalmazott szoftverek.

3.5.3.1. Alice

Az Alice (Carnegie Mellon University, 2019) – a blokk-alapú vizuális nyelvek táborába tartozva – segítségével a tanulók interaktív 3D animációkat tudnak készíteni. Érdekesség, hogy az Alice eseményeket is felsorakoztat az eszköztárában, mégsem összekeverendő az esemény-utasítás alapú vizuális programozással, ugyanis a vizuális kód nagy része a blokk-alapú megközelítést követi. Felépítését tekintve bevezető nyelvként szolgál az objektum orientált programozás világába (Fincher et al., 2010). Az informatikaoktatás számos szintjén alkalmazzák, mint bevezető programozási nyelvet. Korábbi kutatások alapján a tanulók érdekesnek és szórakoztatónak találják az Alice-el készített munkákat. A szoftver pedig hasznosnak, könnyen érthetőnek és kezdők számára gyorsan tanulhatónak bizonyult (Sykes, 2007).

3.5.3.2. Lego Mindstorms

A Lego robotok programozására használt Lego Mindstorms (Lego, 2019) környezet az esemény-utasítás alapú vizuális programozás egy variánsát alkalmazza. Eszközkészletét tekintve kezdőknek ajánlott, de haladó felhasználók szöveg alapú programnyelvek segítségével is írhatnak kódrészleteket (C++ és Java). Fontos szempont, hogy a tanulói munkát azonnali, fizikai visszajelzés követi. Alkalmazási területe nem szorítkozik kizárólag a közoktatásra, jelen van a felsőoktatásban is (Klassner & Anderson, 2003). Széleskörű bevezetése azonban több problémába is ütközik. Egy kutatás kimutatta, hogy a Lego Mindstormst használó tanulók nem teljesítettek jobban, mint azok, akik hagyományos programozási környezeteken keresztül tanulták az érintett témakört. Továbbá, a robotok használata nem motiválta a tanulókat arra, hogy a felsőoktatás magasabb szintjein folytassák tanulmányaikat (Cliburn, 2006). Ezen túlmenően a robotprogramozás magas eszközigénnyel rendelkezik. Figyelembe véve a készletek árait, a Lego robotok beszerzése megfelelő mennyiségben a legtöbb közoktatási intézmény számára komoly anyagi kihívást jelentő beruházás.

3.5.3.3. Scratch

Az informatikaoktatás bármely szintjén találkozhatunk a blokk-alapú vizuális programozást használó Scratch (Lifelong Kindergarten Group, 2019; Resnick et al., 2009) környezettel. Kialakítását tekintve ez a környezet is a kezdő programozók táborát célozza meg, külön figyelve a fiatal korosztály (8–16 éves) azon tagjaira, akik nem tudták magukat szoftverfejlesztőként elképzelni mielőtt megismerkedtek a Scratchhel (Resnick et al., 2009). Számos oktatásra kifejlesztett funkciót tartalmaz, például a projektek megoszthatósága, vagy a forráskódjuk publikus elérhetősége. Bár a tanulók könnyen kezelhetőnek találták a környezetet, számos tanulmány fókuszál a munkafolyamatában és hatékonyságában rejlő problémákra. Egy általános iskolában kimutatták, hogy a Scratch használata nem eredményezett magasabb problémamegoldási készségeket szemben a hagyományos módszerekkel, 5. évfolyamos tanulók esetében (Kalelioglu & Gülbahar, 2014). A környezeten keresztül a tanulók hajlamosak rossz programozási gyakorlatokat kialakítani és követni: a feladat megoldásához látszólag szükséges vagy kapcsolódó összes építőelemet hozzáadni a projekthez azok funkciójának vagy felhasználási módjainak átgondolása nélkül. Ugyancsak előfordulnak logikai összefüggések nélküli túlságosan lebontott elemek is a tanulói munkákban (Meerbaum-Salant et al., 2011). További probléma, hogy a Scratch nem inicializálja újra a változók értékét a projekt futtatásai között. Ez rossz inicializációs szokások kialakulásához vezet, valamint megnehezíti a későbbi tudástranszfert más környezetek között (Franklin et al., 2016).

3.5.3.4. Kodu Game Lab

A Kodu Game Lab (Microsoft Research, 2019) célja, hogy a felhasználók multimédia elemekben gazdag egyszerű játékokat és történeteket hozhassanak létre, a többszenzoros ingerlést szem előtt tartva. A környezet egy egyszerűsített eseményutasítás alapú vizuális nyelvre támaszkodik a projektek logikájának felépítésében. Fiatal korosztályt céloz meg és tervezéséből adódóan – maga a környezet is olyan hatást kelt, mintha egy játékkal játszana a tanuló – az egyéni felfedezésen alapuló

tanulási folyamatot támogatja (Fowler, 2012). A vizuális nyelvének használatán keresztül tanít programozási koncepciókat és alkalmas lehet a hagyományos, szöveg alapú programozási nyelvekkel történő munka megalapozására (Stolee & Fristoe, 2011).

A fentebb ismertetett környezetekben közös, hogy mindegyiket oktatási céllal hozták létre és ezáltal nem, vagy csak korlátozottan alkalmasak önálló szoftverek fejlesztésére. Tapasztalataink és vizsgálataink alapján a közoktatásban tanuló diákok ezeket a szoftvereket sokszor gyerekesnek találják, ami megnehezíti a bevezetésüket idősebb korosztályok számára. Munkánk során egy olyan vizuális programozásra alapuló fejlesztőkörnyezet lehetőségeit kutattuk, amely elsődlegesen szoftverfejlesztési céllal lett létrehozva, azonban egyszerű használatából eredően alkalmas lehet oktatási alkalmazásra is. Célunk az volt, hogy egy olyan környezettel ismertessük meg a tanulókat, amely magába foglalja annak a lehetőségét, hogy összetett, piacra készített alkalmazások is készíthetők a segítségével.

3.5.3.5. Construct 3

Választásunk a HTML5 technológiára alapuló, esemény-utasítás és viselkedés alapú vizuális programozási lehetőségeket támogató Construct 3 (Scirra, 2019a) környezetre esett. A Construct 3 alkalmas 2D játékok és multimédia web alkalmazások fejlesztésére és kialakítását tekintve általános-célú megközelítést követ mind a vizuális nyelv, mind az utasításkészletét tekintve. Választásunkat indokolják a környezet fejlesztőinek törekvései, hogy saját implementációjuk az esemény-utasítás alapú vizuális kódolásnak a lehető legkevesebb akadályt és korlátozást gördítse a felhasználók útjába. További indok, hogy a tanulók a gyors fejlesztési folyamatnak köszönhetően már az első tanórán interaktív, vizuálisan is látványos alkalmazásokat tudnak készíteni. A környezet rendelkezik minden szükséges eszközzel az alapvető programozási koncepciók tanítására.

A Construct 3 széleskörűen dokumentált környezet, amely elősegíti a tanórákba történő integrálását (Alexander, 2016; Scirra, 2019b, 2019c, 2019d; ScirraVideos, 2019). Az oktatási folyamatok további támogatására a környezet lehetőséget biztosít

– hasonlóan a Scratchhez – a tanulói munkák közös online felületre való feltöltésére és egyszerű megosztására. A program korlátozásokkal ingyenesen elérhető, azonban annak ellenére, hogy a Construct 3 lehetőségeinek teljes kiaknázásához licenszet kell vásárolni, az ingyenes verzió oktatásban való felhasználása hivatalosan is engedélyezett.

Kiemelnénk, hogy a szoftverrel való munkánk kezdetekor annak egy korábbi, Windows operációs rendszerre elérhető verziójával, a Construct 2-vel dolgoztunk. A böngészőprogramokban futó, teljesen újraírt Construct 3-at annak stabil verzióinak megjelenésekor kezdtük el alkalmazni.

4. Feladatmegoldás a gyakorlatban

Az informatikaoktatásban jelen lévő hagyományos és algoritmikus szemléletű módszerek megértését elősegíti a velük végzett feladatmegoldások áttekintése. A következőkben bemutatunk néhány, válogatott feladatot témakörökre bontva. A feladatok kiválasztásánál törekedtünk arra, hogy azok reálisan tükrözzék az informatikaoktatásban gyakran előforduló problémamegoldási megközelítéseket, a hagyományos, és a 3.5 fejezetben bemutatott módszertanokhoz kapcsolódóan.

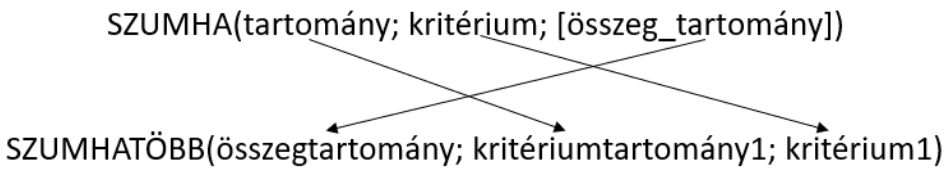
4.1. Hagyományos módszerek

A korábban ismertetett felületi megközelítések azonosíthatók a feladatokban is. A tanulók által megoldott problémák sok esetben nem tükrözik a valós életben felmerülő informatikai elvárásokat. Gyakori a lineáris instrukciókat követő, „szakácskönyv" jellegű feladatok megjelenése, amelyek teljesítéséhez a tanulóknak elegendő lépésről-lépésre végrehajtani a kért műveleteket tényleges gondolkodás nélkül. Egy másik, magasabb gondolkodási és feladatmegoldási szintet tükröző megközelítés a minta alapján történő feladatvégzés. A gyakorlatban azonban legtöbbször maguk az elkészítendő mintadokumentumok is számos hibát tartalmaznak, amelyek nem szolgáltatnak jó alapot az algoritmikus készség és a számítógépes gondolkodás fejlesztésére, csupán a bennük megjelenő, „barkácsolás" elvét követő szerkesztési tudásszint átadására alkalmasak.

4.1.1. Táblázatkezelés

A táblázatkezelés oktatásának egyik központi eleme a táblázatkezelői képletek és függvények használata. A tanulók már a témakör első előfordulásakor megismerkednek a probléma-specifikus függvényekkel. Ezzel párhuzamosan, a táblázatkezelői képletek írása mellett hamar bevezetésre kerül az AutoSzum vagy a Függvény beszúrása varázsló használata. A varázslóval alkotott (legtöbbször egyszintű) képletek mögöttes működése sokszor rejtve van a felhasználó előtt, annak mélyebb átlátását nem szorgalmazza és könnyíti meg a felület.

További problémaként merül fel, hogy a tanulóknak el kell sajátítaniuk ezeknek a probléma-specifikus függvényeknek a neveit és (sokszor logikátlan tervezésű) argumentumait (Walkenbach, 2010) (2. ábra).



2. ábra: A hasonló funkciót betöltő probléma-specifikus függvények argumentumsorrendjében fellelhető következetlenségek a SZUMHA() és SZUMHATÖBB() függvények példáin keresztül.

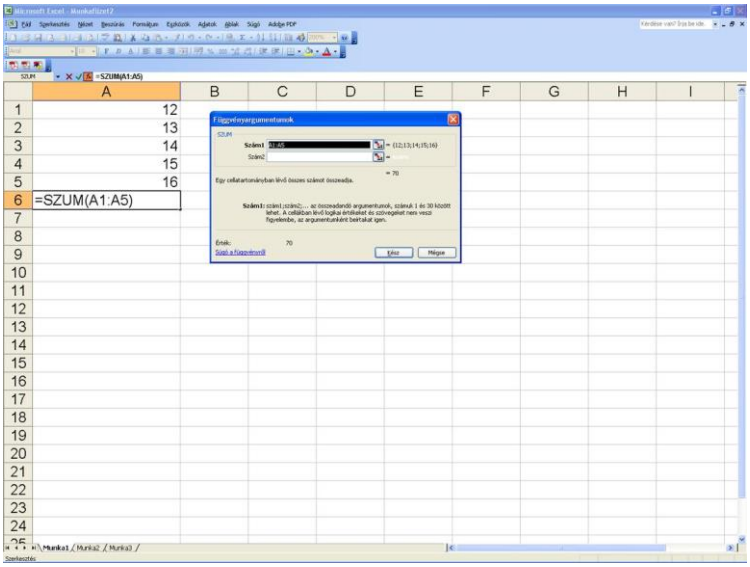
Figyelembe véve a táblázatkezelő szoftverekbe beépített függvénybázis méretéből adódó variációkat, ez indokolatlan terhet jelent a tanulók számára. A Microsoft Excel programban közel 600 beépített függvény található (Microsoft, 2019a), valamint a magyar táblázatkezeléssel foglalkozó könyvek közel 200 probléma-specifikus függvényt említenek (Csernoch et al., 2014a). Ezen túlmenően egy felmérésben résztvevő tanulók eredményeit összegezve 99 különböző táblázatkezelői függvény gyűlt össze arra a kérdésre, hogy nevezzék meg a 15 legfontosabb függvényt (Biró & Csernoch, 2013; Biró et al., 2015). A probléma-specifikus függvények alkalmazásának nehézségeire jó példaként szolgál a DARABTELI() függvény. Ellentétben az eldöntendő kérdések táblázatkezelői szintaktikájával, az egyenlőtlenségek vizsgálatának leírása kizárólag összefűzés segítségével lehetséges (3. ábra).

```
=DARABTELI(D2:D101;">"&500)
{=SZUM(HA(D2:D101>500;1))}
```

3. ábra: A feltételes megszámlálást végző DARABTELI() függvény (felül) és egy feltételes megszámlálást végző összetett tömbképlet (alul).

További, hasonló kategóriába tartozó példa, hogy a diákok mindenféle kontextus nélküli, olykor fiktív adatokon végeznek célt nélkülöző számításokat. Az ilyen feladatoknak a függvény működésének bemutatásán túl más funkciója nincs és ebből adódóan nem is alkalmasak a tudáselemek mélyebb rögzülésének elősegítésére. Ezt

a tendenciát jól szemlélteti a 4. ábra. A példatáblázatban minimális adatmennyiséggel (öt rekord) dolgoznak a diákok, melyek jelentése ismeretlen számunkra és a táblázat elemzéséből nem is kideríthető.



4. ábra: Példafeladat a SZUM() függvény használatának bemutatására (Sulinet, 2019).

A függvények használatán túlmenően a hagyományos megközelítés nagy hangsúlyt fektet a táblázatok formázására is. A feladatok vagy a korábban már ismertetett „szakácskönyv” jellegű utasításokon keresztül, vagy mintákat mutatva adnak formázási feladatokat a diákoknak. Ezek a formázások azonban sok esetben nem a táblázatba foglalt adatok helyes megjelenítését szolgálják, vagy az algoritmusalapú feltételes formázást gyakoroltatják. Ehelyett a diákoknak „színes-szagos” táblázatokat kell elkészíteniük, melyek megformázásának egyetlen célja a formázási funkciók felületen elfoglalt helyének és működésnek bemutatása.

		SZOMBAT			VASÁRNAP				12%
		Krumpli	Hagyma	Alma	Krumpli	Hagyma	Alma	Összbevétel	ÁFA
PIAC	Náncsi néni	7kg	6kg	7kg	11kg	8kg	10kg	5 646 Ft	678 Ft
	Kutiné	4kg	10kg	2kg	11kg	8kg	8kg	5 315 Ft	638 Ft
	Pofa néne	8kg	11kg	1kg	6kg	8kg	10kg	4 928 Ft	591 Ft
	Zsó néni	6kg	5kg	4kg	11kg	1kg	1kg	3 320 Ft	398 Ft
	Molli mama	8kg	9kg	3kg	9kg	3kg	11kg	4 506 Ft	541 Ft

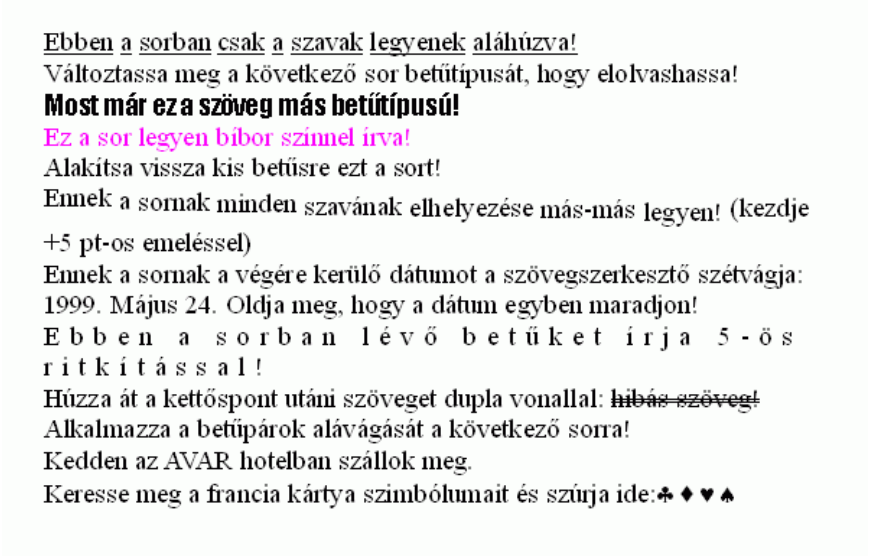
ÁRAK	Náncsi néni	96 Ft	139 Ft	116 Ft
	Kutiné	123 Ft	135 Ft	104 Ft
	Pofa néne	104 Ft	115 Ft	117 Ft
	Zsó néni	119 Ft	127 Ft	107 Ft
	Molli mama	100 Ft	116 Ft	101 Ft

5. ábra: Példafeladat egy formázott táblázatra (Nytta, 2019a).

A példafeladat (5. ábra) jól illusztrálja az olyan formázási gyakorlatokat, amelyek megnehezítik a táblázatba foglalt adatok olvasását és elemzését (betűszín, szöveg írásának iránya, félkövér formázás, csupa nagybetűs írásmód és a „kg” mértékegységgel ellátott adatok középre igazítása).

4.1.2. Szövegszerkesztés

A felületi elemek és beépített funkciók használata a szövegszerkesztés témakörben még nagyobb hangsúlyt kap. Ezeknek a funkcióknak azonban hátránya, hogy nem segítik a tudatos dokumentumtervezést és a tanulók megfelelő instrukciók hiányában az aktuális formázási, szerkesztési lépéseken túl nem is számolnak azok dokumentumra kifejtett hatásaival. Emellett számos kényelmi, felületi funkció szoftverkörnyezet- és vagy verziófüggő, amely megnehezíti (vagy olykor ellehetetleníti) alkalmazásukat a megszokottól eltérő munkakörnyezetben. Hasonlóan, ahogy ezeknek a funkcióknak az alkalmazása nem igényli a dokumentum tudatos megtervezését, a dokumentum szerkezetét jól tükröző nem nyomtatható karakterek csak ritkán játszanak szerepet a feladatmegoldásban.

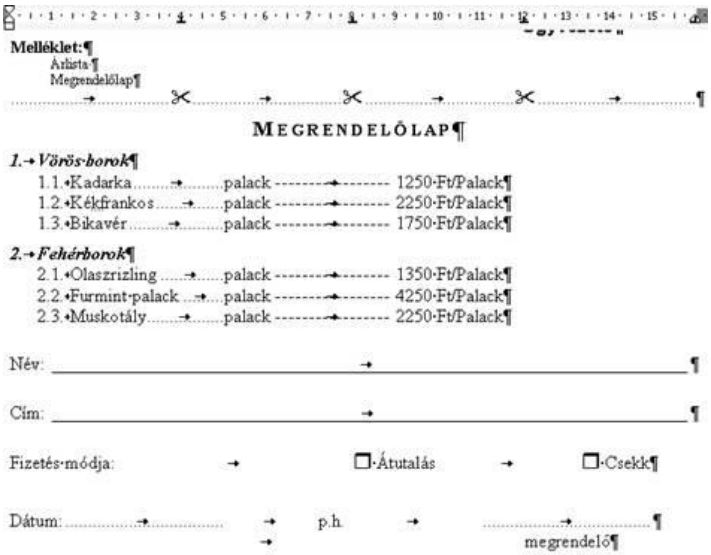

Ebben a sorban csak a szavak legyenek aláhúzva!
Változtassa meg a következő sor betűtípusát, hogy elolvashassa!
Most már ez a szöveg más betűtípusú!
Ez a sor legyen bíbor színnel írva!
Alakítsa vissza kis betűsre ezt a sort!
Ennek a sornak minden szavának elhelyezése más-más legyen! (kezdje +5 pt-os emeléssel)
Ennek a sornak a végére kerülő dátumot a szövegszerkesztő szétvágja: 1999. Május 24. Oldja meg, hogy a dátum egyben maradjon!
E b b e n a s o r b a n l é v ő b e t ű k e t í r j a 5 - ö s r i t k í t á s s a l !
Húzza át a kettőspont utáni szöveget dupla vonallal: ~~hibás szöveg!~~
Alkalmazza a betűpárok alávágását a következő sorra!
Kedden az AVAR hotelban szállok meg.
Keresse meg a francia kártya szimbólumait és szúrja ide:♣ ♦ ♥ ♠

6. ábra: Példa egy karakterformázásokat tanító szövegszerkesztési feladatra (Nytta, 2019b).

Az 6. ábra jól illusztrálja a valódi cél nélküli szövegszerkesztési feladatok kategóriáját, mely nem tartalmaz összefüggéseket, sem gondolkodtató tudáselemeket.

Ezen túlmenően a példafeladat még a „barkácsolás” gyakorlatát is erősíti, ugyanis számos – a 3.5.2.1-es fejezetben taglalt – hibát is magába foglal, utasítva a tanulókat, hogy maguk is elkövessék azokat.

A 7. ábra egy olyan feladat végtermékét mutatja be, amelynek elkészítésekor már tudatos tabulátorhasználatot tanulnak a diákok. Azonban a dokumentum nem megfelelő eszközöket használ a tartalom elrendezésére, amely következében ugyanúgy hibás feladatnak minősül attól függetlenül, hogy az alkalmazott eszközt szakszerűen építette be. A szemléltetett dokumentumban (7. ábra) a borok adatainak felsorolása és tabulátorral történő tagolása egy klasszikus példa a táblázatos elrendezés helyett tabulátorok használatára. Hasonló példaként szolgál még a dokumentum alsó részén látható űrlap, melynek kialakítása tabulátorok helyett a szövegszerkesztő program beépített vezérlőinek használatával lenne indokolt.


Melléklet:
Árlista
Megrendelőlap

MEGRENDELŐLAP

1. Vörös borok
1.1. Kadarka ..... palack ------- 1250 Ft/Palack
1.2. Kékfrankos ..... palack ------- 2250 Ft/Palack
1.3. Bikavér ..... palack ------- 1750 Ft/Palack

2. Fehérborok
2.1. Olaszrizling ..... palack ------- 1350 Ft/Palack
2.2. Furmint palack ..... palack ------- 4250 Ft/Palack
2.3. Muskotály ..... palack ------- 2250 Ft/Palack

Név: _____

Cím: _____

Fizetés módja: ☐ Átutalás ☐ Csekk

Dátum: ..... p.h. ..... megrendelő

7. ábra: Példa egy tabulátorokat tanító szövegszerkesztés feladat végtermékére (Béres, 2006).

Akárcsak a táblázatkezelés esetében, a szövegszerkesztés produktumai között is gyakoriak az úgynevezett „színes-szagos” dokumentumok (8. ábra). A végtermék, bár kétségkívül figyelemfelkeltő, nehezen olvasható és egyetlen üzenetet közvetít tisztán: a felhasználó nem tudja mit csinál a szövegszerkesztő programban. Fontosnak tartjuk kiemelni, hogy ezek a dokumentumok nem a szemléltetés kedvéért kiragadott példák, hanem valós állapotokat tükröznek. Az informatikaoktatásban és az

interneten számos hasonló dokumentummal lehet találkozni akár az oktatásra szánt feladatok, akár a tanulói, vagy végfelhasználói produktumok halmazait vizsgáljuk.

8. ábra: Példa egy sakkversenyt hirdető dokumentumra (részlet) (Száva, 2010).

A hagyományos, felületi alapú megközelítéssel kapcsolatosan számos további példát és esetet ismertethetnénk, azonban ennek az értekezésnek nem célja a szövegszerkesztési hibák előfordulásainak és variációinak teljességre törekvő bemutatása. Úgy véljük, hogy a fentiekben bemutatott példák rávilágítanak a tradicionális feladatmegoldási megközelítés problémáira.

4.1.3. Programozás

Hasonlóan az alkalmazott eszközökhöz, a témakörbe foglalt feladatok azok szintjét, célját és komplexitását tekintve meglehetősen változatos formában vannak jelen az informatikaoktatásban. Az szöveg alapú programnyelvek mellett az EPL-eken keresztüli (3.5.3 fejezet) feladatmegoldás is szerepel, de egyes intézményekben akár az Excel VBA (Visual Basic for Applications) is lehet a programozásoktatás fő eszköze.

Akárcsak az informatika egyéb témakörei, a programozás is ki van téve a hibás feladatok, a nem megfelelő logika alapján felépített algoritmusok és forráskódok veszélyeinek. Az alábbi kódrészlet (9. ábra) jól szemlélteti a táblázatkezelés és szövegkezelés témaköröknél már ismertetett megközelítést. A feladat kiragadja a nyelv egy

funkcióját és valódi feladat megoldása nélkül mutatja be azt, kódrészleteken keresztül.

```
string s5 = "tetszoleges";
cout << "s5 elso karaktere: " << s5[0] << endl;
s5[1] = '3';
s5[s5.length()-1] = '5';
cout << "s5 modositott erteke: " + s5 << endl;
```

9. ábra: Példa forráskód string műveletekre egy C++ nyelvet oktató programozás tananyagból (Christo161, 2016).

Hasonló elvet követ a következő forráskód is (10. ábra), melyben a StreamWrtier osztály alkalmazási módja kerül bemutatásra. A forráskód számos hibát tartalmaz: az elején nem használt névterek jelennek meg, az osztály neve nem utal annak funkciójára, a beolvasott fájl elnevezése és a fájlba írást végző objektum neve ékezetes karaktereket tartalmaznak, és az utóbbi nem utal az objektum funkciójára, a kiírás utasítása cél nélküli és helyesírási hibát tartalmazó példaszöveget ír ki a fájlba, illetve hiányoznak a megértést elősegítő megjegyzések.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;

namespace ConsoleApplication11
{
    class Program_TR
    {
        static void Main(string[] args)
        {
            string fajl_TR = @"C:\\téikszté.txt";
            StreamWriter téikszté_TR = new StreamWriter(fajl_TR, false, Encoding.Default);
            téikszté_TR.WriteLine("Ki iras helye");
            téikszté_TR.Close();
            Console.ReadLine();
        }
    }
}
```

10. ábra: Példaprogram a StreamWriter osztály használatára C# nyelven (Tóth, 2017).

Az oktatási céllal kidolgozott környezetek sem mentesek a hibamentes feladatoktól. A 11. ábra egy olyan Scratch feladat forráskódját mutatja be, amely a 100-150 intervallumba foglalt öttel osztható egész számokat jeleníti meg. A forráskód azonban több ponton is a szokványostól eltérő megközelítést alkalmaz: az algoritmus kü-

lön indoklás nélkül csak az „o” billentyű lenyomásakor fut le, illetve egy végtelenített ciklust használ a számok kiíratására, melyből egy külön feltételvizsgálattal lép ki. Ez értelmezhető hátultesztelő ciklusnak is, azonban a feladatleírás (Burcsi, 2017) semmilyen utalást nem tesz erre a tudáselemre.

11. ábra: Példaprogram Scratch környezetben az öttel osztható háromjegyű számok kiíratására a 100–150 tartományban (Burcsi, 2017).

A fejezetben bemutatott feladatok azzal a céllal kerültek kiválasztásra, hogy rávilágítsanak az informatikaoktatásban jelenlévő hibás feladatmegoldási stratégiákra. Természetesen nem állítjuk, hogy a bemutatott feladatok és hibák a teljes informatikaoktatás világát tükrözik, ugyanis találkozhatunk kiváló minőségű tananyagokkal, feladatokkal, ahogy szakmailag precíz és kifogástalan informatika tanárokkal is. Ettől függetlenül tapasztalataink szerint a hibás megközelítések nagy számmal vannak jelen az informatikaoktatás gyakorlatában, melyek átgondolása, kijavítása és átalakítása az oktatás valamennyi szereplőjének érdeke.

4.2. Nem hagyományos módszerek

Az általunk vizsgált nem hagyományos módszerek algoritmikus szemléletet tükröznek, amely áthatja a feladatok kidolgozásának, megoldásának és ellenőrzésének folyamatait. A feladatok tágabb kontextusban gondolkodnak, mint a hagyományos megközelítések esetében ismertetett példák, a tudáselemek alkalmazhatóságát, újbóli előhívását és mély rögzülését támogatva.

4.2.1. Spreadsheet Lego

A Sprego módszertannal történő feladatmegoldás során a tanulók minden új adattáblával való megismerkedéskor elvégzik tanári instrukciók mellett annak előkészítését:

– a táblázatba foglalt adatok elemzése,
– másolat készítése a munkalapról,
– a munkavégzéshez felesleges sorok és oszlopok elrejtése,
– ablaktábla rögzítése.

A táblázat előkészítését követően a tanulók rátérnek a táblázatkezelői problémák megoldására. A folyamat bemutatását a feltételes megszámlálás algoritmusán keresztül végezzük.

	A	B	C	D	E	F	G
1	Year	NameOfTheProperty	Country	Type	Region	Property_ha	ID
3	1978	City of Quito	EC	C	LAC	320	2
4	1978	Galápagos Islands	EC	N	LAC	14066514	1
5	1978	Historic Centre of Kraków	PL	C	EUR	150	29
6	1978	Island of Gorée	SN	C	AFR	0	26
7	1978	L'Anse aux Meadows National Historic Site	CA	C	EUR	7991	4
8	1978	Mesa Verde National Park	US	C	EUR	21043	27
9	1978	Nahanni National Park #	CA	N	EUR	47656	24
1201	2019	Water Management System of Augsburg	DE	C	EUR	113	1580
1202	2019	Writing-on-Stone / Áísínai'pi	CA	C	EUR	1106	1597

12. ábra: A Sprego feladatmegoldás során használt, előkészített World Heritage táblázat (UNESCO World Heritage Centre, 2019).

A feltételes megszámlálás Sprego megoldásának bemutatására a következő feladatot végezzük el a World Heritage táblázatban (12. ábra): Adjunk meg egy régiót (az E mezőben látható formátumban) a H2 cellában. Ezt követően számoljuk ki a megadott régióban lévő világörökségek számát. A megoldás algoritmusa a következő lépéseket követi:

1. Eldöntendő kérdést fogalmazunk meg, amely segítségével megvizsgáljuk az egyes világörökségekre vonatkozóan, hogy az adott régióban található-e. Képlet: {=E2:E1202=H2}

2. Az eldöntendő kérdés által visszaadott IGAZ/HAMIS értékek közül megjelöljük az IGAZ értékeket egyesekkel. Képlet: {=HA(E2:E1202=H2;1)}
3. Összeadjuk az egyeseket. Képlet: {=SZUM(HA(E2:E1202=H2;1))}

A lépéseket és a hozzájuk tartozó példa képleteket végig követve jól látható a bentről kifele építkezés elve, ahol minden következő lépése az algoritmusnak az előző lépés outputjára épít.

A feltételes megszámlálás során használt algoritmus elsajátítása kulcsfontosságú szerepet tölt be a táblázatkezelői problémák megoldásában. Erre alapozva, illetve problémakontextustól függően az algoritmus 2. lépésének outputját és a 3. lépés műveletét módosítva számos probléma-specifikus függvény kiváltható (4. táblázat).

4. táblázat: Azonos alapalgoritmusra épülő táblázatkezelői képletek a Sprego módszertannal felépítve és azok probléma-specifikus függvénnyel előállított megfelelői.

Sprego képlet	Probléma-specifikus függvény
{=SZUM(HA(E2:E1202<>"";1))}	=DARAB2(E2:E1202)
{=SZUM(HA(F2:F1202>H3;1))}	=DARABTELI(F2:F1202;">"&H3)
{=SZUM(HA(F2:F1202>H3;F2:F1202))}	=SZUMHA(F2:F1202;">"&H3)
{=ÁTLAG(HA(F2:F1202>H3;F2:F1202))}	=ÁTLAGHA(F2:F1202;">"&H3)
{=MAX(HA(F2:F1202>H3;F2:F1202))}	=MAXHA(F2:F1202;F2:F1202;">"&H3)
{=MIN(HA(F2:F1202>H3;F2:F1202))}	=MINHA(F2:F1202;F2:F1202;">"&H3)

4.2.1.1. Unplugged eszközök a feladatmegoldásban

Az unplugged eszközök mindegyike az algoritmus egy-egy lépését írja le. Hajtogatott papírcsónakok esetében első lépésként a tanári kérdések megválaszolását követően a csónakok belsejébe kerül behelyezésre vagy írásra az adott lépés inputja. Ezt követi a csónakok oldalára vezetett lépés leírása természetes nyelven megfogalmazva, majd a táblázatkezelő programban az adott lépés elvégzése (kódolása). Utolsó lépésként a csónakok oldalára az adott lépés által visszaadott értékek, az outputok kerülnek felírásra.

A csónakok (és más unplugged eszközök) különböző méretei és színei a lépések felépítésének megértését szolgálják. Minden feladat megoldásához a lépések számával megegyező, eltérő méretű és színű eszközt kell biztosítani a tanulók számára.

4.2.1.2. Oktatószoftverek a feladatmegoldásban

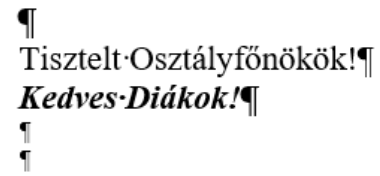
A Sprego módszertanban a feladatmegoldást a korábban bemutatott (3.5.1.2 fejezet) 2D oktatóprogram (Csapó & Sebestyén, 2017; Sebestyén & Csapó, 2018) kíséri. A feltételes műveletvégzés és lineáris keresés algoritmusainak ismétlésére és áttekintésére azok első előfordulásakor, valamint az azt követő tanórák első pár percében célszerű a tanulókat folyamatosan kérdezve bevezetni.

Az oktatószoftver tanórai keretek közötti alkalmazásának tapasztalatai annak pozitív fogadtatását támasztották alá (Csapó & Sebestyén, 2017). A grafikus felület megragadta a tanulók figyelmét, azt érdekesnek és motiválónak találták korosztálytól függetlenül. Az érintett osztályokban a diákok aktívabbak voltak és gyakrabban válaszoltak a tanári kérdésekre. Oktatói oldalról megközelítve az alkalmazás használata lehetővé tette, hogy a Sprego órák dinamikusabban teljenek kevesebb szükséges magyarázattal és ismétléssel, illetve, hogy a tanulók komplexebb problémák megoldásán is dolgozhassanak. Hasonlóan pozitív fogadtatásban részesült az alkalmazás informatika tanár szakos hallgatók körében. A hallgatók befogadták az oktatószoftvert saját eszköztárjukba, valamint segítette őket a Sprego módszertan elsajátításában. A hallgatók véleménye alapján a szoftver segítőkész és motiváló hatású, illetve megkönnyíti az összetett képletek mögött meghúzódó számítógépes logika megértését. Kiemelnénk, hogy ezek az eredmények megfigyeléseken, visszajelzéseken és tapasztalatokon alapulnak. Az oktatószoftver hatékonyságának kontroll csoporttal történő vizsgálata és statisztikai kiértékelése folyamatban van.

4.2.2. Error Recognition Model

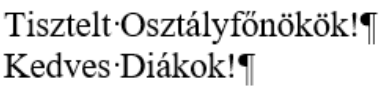
Az ERM módszertanban a feladatmegoldás első lépése, hasonlóan a Spregohoz, a dokumentum elemzése. A tanulók a munkavégzés alapjául szolgáló szöveges dokumentumot tanári irányítás mellett először nyomtatott, majd digitális formában is áttekintik. Az elemzés folyamata szorosan összekapcsolódik a dokumentum kijavításával, melyet a kezdeti megbeszélés után a tanulók folyamatosan végeznek, továbbra is tanári irányítás és ellenőrzés mellett. Az elemzés során minden felmerülő

hiba esetén fontos annak tisztázása, hogy miért tekintjük hibásnak az adott szövegrészletet. Amennyiben indokolt, a hiba gyakorlati következményeinek bemutatásával szemléltetni is kell a hibajelenséget. A hiba kijavítása csak azután kezdődhet az ERM-ben, miután a tanulók megértették annak szükségességét.



13. ábra: Hibás szövegrészlet (megszólítás) a Mozirajongók (Karakas, 2003) dokumentumból.

A 13. ábra jól szemléltet két gyakori szövegszerkesztési hibát is: A megszólítás alatt és felett üres bekezdések találhatók térköz használata helyett, valamint a második megszólítás dupla kiemeléssel rendelkezik: félkövér és dőlt. Emellett még megemlíthetjük, hogy az üres bekezdések eltérő betűmérettel kerültek megformázásra. Ezeknek a hibajelenségeknek a megbeszélését követően azok besorolásra kerülnek a megfelelő hibakategóriákba (3.5.2.1 fejezet), melyet kijavításuk követ (14. ábra). A javítás és a későbbi megbeszélések során nagy figyelmet kell fordítani arra, hogy a dokumentum felépítése konzekvens legyen, melybe beletartozik, hogy a megegyező szerkezeti elemek azonos formázást kapjanak.



14. ábra: Kijavított szövegrészlet (megszólítás) a Mozirajongók (Karakas, 2003) dokumentumból.

Egy másik gyakori hibajelenség a felsorolások kézzel történő számozása (15. ábra). Ennek gyakorlati hatása könnyen szemléltethető, elegendő egy új pontot beszúrni a lista elejére. A hibás felsorolás oktatása mellett a szövegrészlet további hibákat és tudáselemeket is tartalmaz: a felsorolás feletti bekezdés tartalma alá van húzva, a felsorolás egyes pontjaiban lévő felszólító mondatok mondatvégi írásjele hibás, a felsorolás 3. pontjában két mondat van egybe írva, a 4. pontban az e-mail

szó szintaktikailag hibásan van írva, az e-mail- és postacímek többszörösen ki vannak emelve, illetve felesleges szóköz található a bekezdés végén.

A feladat:¶
1. Vesd be magad kedvenc mozidba, és nézd meg bármely május 8-tól játszott filmet.¶
2. Húzz haza és gondold végig a látottakat.¶
3. Ragadj ceruzát, tollat, billentyűzetet vagy egy málnaszörpös szívószálat, és vésd papírra véleményed, érzéseid, rajongj vagy állj bosszút az unalmas percekért, amit a film okozott.¶
4. Alkotásod küldd el emilen az ***info@filmrajongo.****hu* e-postacímre, vagy add postára az ***1410 Budapest Pf. 119.*** címre szóló borítékban. ¶

15. ábra: Hibás szövegrészlet (felsorolás) a Mozirajongók (Karakas, 2003) dokumentumból.

A fenti hibajelenségek csupán kiragadott példák, melyek az ERM módszerrel történő feladatmegoldás sokszínűségét mutatják be. A hibák kijavításán keresztül (16. ábra) a tanulók fokozatosan előre haladva ismerkednek meg a szövegszerkesztő környezet funkcióival anélkül, hogy azok kapnák a hangsúlyt a probléma megoldásában.

A feladat:¶
1. Vesd be magad kedvenc mozidba, és nézd meg bármely május 8-tól játszott filmet!¶
2. Húzz haza és gondold végig a látottakat!¶
3. Ragadj ceruzát, tollat, billentyűzetet vagy egy málnaszörpös szívószálat, és vésd papírra véleményed és érzéseid! Rajongj vagy állj bosszút az unalmas percekért, amit a film okozott!¶
4. Alkotásod küldd el e-mailen az info@filmrajongo.hu e-postacímre, vagy add postára az 1410 Budapest, Pf. 119. címre szóló borítékban!¶

16. ábra: Kijavított szövegrészlet (felsorolás) a Mozirajongók (Karakas, 2003) dokumentumból.

4.2.2.1. Unplugged eszközök a feladatmegoldásban

A tanulók a módszertannal történő munkavégzés elején megkapják a dokumentumot nyomtatott formában unplugged eszközként. A tanulók az elemzési és javítási folyamatokkal párhuzamosan végzik a hibák megjelölését és megnevezését a lapokon a korábban ismertetett módon (3.5.2.2 fejezet) (Függelék 15.4). A diákok a nyomtatott formát beragasztják a füzetükbe, melynek jelöléseit a dokumentumban való előrehaladás során folyamatosan bővítik, valamint referenciaként is felhasználják későbbi hasonló feladatok megoldásához.

4.2.3. Vizuális programozás

A vizuális programozással történő feladatmegoldás formája nagy mértékben függ a választott fejlesztőkörnyezet sajátosságaitól és a programozási nyelv felépítésétől.

Ebben a fejezetben a Construct 3 környezettel történő feladatmegoldási folyamatot ismertetjük, melyhez nélkülözhetetlen a projektek építőelemeinek áttekintése:

– Objektumok: Más szavakkal: szereplők. Azon elemei a projektnek, amelyek meghatározzák a rendelkezésre álló utasításkészletet.
– Elrendezések: A vizuális megjelenítés felépítésének eszközei. Az objektumokat a projekt elrendezésein elhelyezve alakítható ki a felhasználó által látott felület.
– Viselkedések: Előre megírt kódrészletek, melyeket objektumokhoz kapcsolva gyorsan implementálhatók alapvető funkciók, valamint kiegészíthető azok utasításkészlete.
– Eseménylapok: Az esemény-utasítás alapú vizuális kódokat tartalmazó elemei a projektnek.

4.2.3.1. Vizuális kód felépítése

A projekt esemény-utasítás alapú kódjának létrehozásához az eseménylapon első lépésként definiálni kell egy eseményblokkot egy, a programhoz korábban hozzáadott objektum utasításkészletéből kiválasztott feltétellel. A következő lépés a blokkhoz kapcsolt utasítások megadása. A kész eseményblokk természetes nyelvi megfogalmazáshoz hasonlatos formában jeleníti meg a benne foglalt feltételeket és utasításokat, azok paramétereivel.

17. ábra: Példa a Construct 3 környezetben létrehozott esemény-utasítás alapú vizuális kódra.

Egy egyszerű példakódot szemléltet a 17. ábra, melyben a „billentyuzet” objektum feltételeként a program a bal kurzormozgató billentyű nyomva tartását figyeli. A feltétel teljesülésekor a karakter objektum balra mozgatása valósul meg annak X tengelyen elfoglalt pozíciójának módosításával.

A programozási alapismeretek egyes elemeinek tanítása kézenfekvő a Construct 3 megközelítésén keresztül: az eseményblokkok és a bennük foglalt elemek végrehajtása szekvenciális sorrendben történik, illetve a feltételek szelekciót valósítanak

meg. Fontos kiemelni azonban, hogy a ciklusok bevezetését és oktatását megnehezíti a környezet, ugyanis minden eseményblokk folyamatos iterációban van a feltételek folyamatos ellenőrzése miatt. A környezet tartalmaz eseményblokkokban deklarálható ciklusokat is (for, for-each egy objektum minden példányára vonatkozóan opcionálisan rendezett végrehajtással, ismétlés N alkalommal, while), azonban ezek beépítése a feladatmegoldás menetébe alternatív megközelítéseket igényel.

A változók kezelését a vizuális nyelv a következő formákban teszi lehetővé:

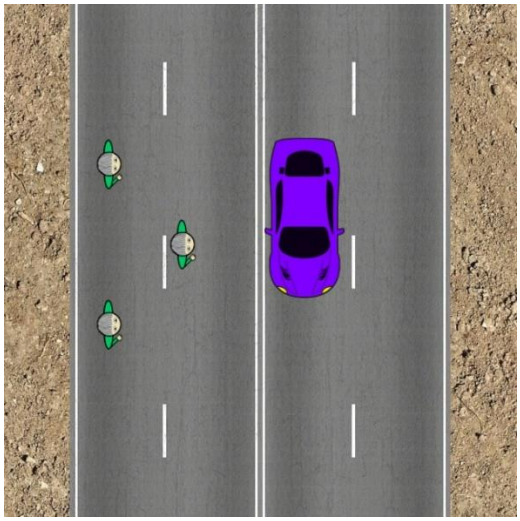
- Globális változók: Elérhetőek az egész projekt számára.
- Lokális változók: Az eseménylapok azon szintjén érthetők el, ahol a deklarálásuk megtörtént.
- Példány változók: Az objektumokhoz kapcsolódnak és az adott objektum minden példánya esetén egyedi értéket vehetnek fel.

A fejlesztőknek lehetőségük van függvények definiálására is a vizuális programkódban, illetve haladó felhasználók JavaScript nyelven írt scriptekkel is kiegészíthetik projektjeiket, bővítve az eredeti környezet funkcionalitását.

4.2.3.2. Feladatok felépítése és kódolása

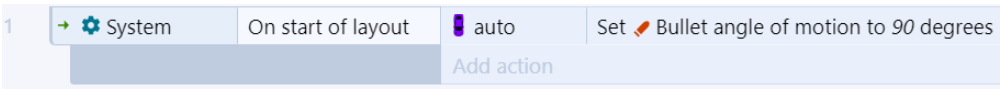
A programozás témakör oktatása a Construct 3 környezetet használva alternatív feladattervezési stratégiát igényel, mint a szöveg alapú programnyelvek esetében. A nyelv és a mögötte meghúzódó játékmotor sajátosságaiból kiindulva fontos, hogy a feladatok a vizuális megjelenítést helyezzék előtérbe, a korosztályhoz igazodott játékosság megtartása mellett. Szintén fontos szempont, hogy a tanulók a lehető legkevesebb lépést követően kézzel fogható produktumot állítsanak elő.

Tapasztalataink szerint a tanulók számára nehézséget jelent a Construct 3 felületének elsajátítása. Ennek megfelelően az első néhány feladat célszerű, ha még nem fókuszál közvetlenül a programozási alapismeretek megtanítására. Szerepük azon túlmenően, hogy a tanulók kézzel foghatónak érezzék munkájuk eredményét, a környezettel való ismerkedés legyen, illetve a projektek elkészítéséhez szükséges elemek és folyamatok áttekintése. Egy ilyen feladat futtatott állapotát mutatja a 18. ábra.



18. ábra: Példafeladat, melyben a tanulók egyenes irányú egyenletes mozgást valósítanak meg.

A feladat kontextusa: három karakter átmegy az úton, miközben fentről egy autó érkezik. A karakterek és az autó is egyenes irányú egyenletes mozgást végeznek, amelyet egy viselkedés hozzárendelésével valósítanak meg a tanulók. Az eseménylapon csak minimális kódolás történik (19. ábra). A tanulók ezt követően kísérleteznek a karakterek mozgását végző viselkedés beállításaival, hogy minden karakter olyan sebességgel mozogjon, hogy elkerülje az érkező autót. A feladat egyszerűségének ellenére számos tudáselemet tartalmaz: projekt létrehozása és beállítása, objektumok beszúrása és elhelyezése, viselkedések objektumhoz rendelése és konfigurálása, projekt futtatása, illetve eseménylapon eseményblokk létrehozása feltétel és utasítás megadásával.



19. ábra: Az egyenes irányú egyenletes mozgást megvalósító példafeladat vizuális kódja.

Miután a tanulók megismerték a környezetet és nem okoz annak alapvető használata tanórai nehézségeket, a későbbi feladatok komplexitása növekedhet (Függelék 15.7).

A feladatok kidolgozása során a tanulók a következő lépéseket követik tanári instrukció mellett:

1. Lépésekre bontás: A kész feladat elemzése és lépésekre bontása, megválaszolva a kérdést: Milyen lépéseken keresztül tudjuk megvalósítani a projektet?
2. Cél kitűzése: A soron következő lépés kiválasztása, majd fókuszba helyezése. Annak megragadása, hogy milyen funkciót szeretnénk implementálni.
3. Algoritmus megtervezése: A cél eléréséhez szükséges algoritmus felépítése, annak input és output értékeinek kiemelésével.
4. Vizuális kód felépítése természetes nyelven: Igazodva az esemény-utasítás alapú vizuális programnyelv szerkezetéhez, az algoritmus kódolásához szükséges feltételek és utasítások természetes nyelven történő megfogalmazása. A lépés megválaszolja a következő kérdéseket: „Minek a bekövetkeztét várom?”, „Mit szeretnénk, hogy történjen annak a bekövetkeztekor?”.
5. Kódolás: Az algoritmus kódolása a Construct 3 eszközkészletét használva.
6. Tesztelés: A projekt futtatása, az algoritmus ellenőrzése és az eredmények megbeszélése.
7. Hibás kód javítása: A kódban felmerülő hibák javítása, ellenőrzése és megbeszélése.

A feladatok megtervezése során az oktatónak mindenképp számolnia kell a vizuális projekteket létrehozó környezetekkel járó többletfeladattal: azok kidolgozása a projekt felépítésén és a benne foglalt algoritmusok megtervezésén túlmenően annak vizuális kialakítását is magába foglalja. Megjegyeznénk, hogy a Construct 3 vizuális programozási eszköztára az ismertetett grafikus projektek elkészítése mellett lehetővé teszi klasszikus programozási feladatok megoldását is.

5. [T1] Spreadsheet Lego hatékonyságvizsgálata

5.1. Alkalmazott módszerek, eszközök és eljárások

A tézisbe foglalt vizsgálat keretein belül a táblázatkezelés oktatására használt módszerek (Spreadsheet Lego – Sprego és hagyományos) hatékonyságát elemeztük. Célunk a tanulók oktatásán és tesztelésén keresztül a Sprego módszertan algoritmikus szemléletének hatékonyságvizsgálata volt középiskolai környezetben.

A mintába egy helyi középiskola három csoportját választottunk ki: két 7. évfolyamos és egy 10. évfolyamos csoportot. Minden csoport az intézmény hat évfolyamos képzésében vett részt. A 10. évfolyamos és az egyik 7. évfolyamos csoport a Sprego módszertan segítségével sajátította el a táblázatkezelés témakört (vizsgálati csoportok), míg a másik 7. évfolyamos csoport a hivatalos középiskolai tankönyvre támaszkodva (Dancsó & Korom, 2013) hagyományos felületi és probléma-specifikus függvényekre épülő megközelítéssel (kontroll csoport). A választott tankönyvet előzetesen elemeztük és nem találtunk olyan tartalmi- és tudáselemeket, amelyek akadályozták volna a kontroll csoportot az előrehaladásban. A könyv teljesen az alapoktól kezdi a témakör feldolgozását alapvető táblázatkezelői függvények használatával és táblák begépelésével. Fontos kiemelni, hogy a két 7. évfolyamos csoport nem rendelkezett előzetes ismeretekkel a táblázatkezelést illetően, míg a 10. évfolyamos csoport a Kerettanterv (OFI, 2013) alapján korábban két évig tanulta a témakört. Első alkalommal 7. évfolyamban találkoztak táblázatkezeléssel (egy óra/hét, 8 héten keresztül), majd ezt követően az előző tanévben, 9. évfolyamban (két óra/hét, 6 héten keresztül). Ezen túlmenően számos 10. évfolyamban tanuló diák ECDL (European Computer Driving License) bizonyítványt szerzett a táblázatkezelés modulból. Ezen információk alapján nem csak azt feltételezhetjük, hogy a tanulók ismerik a témakört, de azt is, hogy olyan megalapozott tudással rendelkeznek belőle, amelyet magabiztosan használnak, és amelyre a tanórák során építeni tudnak.

A vizsgálat alapját képező tanórák számát illetően a két 7. évfolyamos csoport 13 (vizsgálati) és 12 (kontroll) héten keresztül heti egy óra beosztással, a 10. évfolyamos csoport pedig 6 és fél héten keresztül, heti két óra beosztással tanulta a témakört. Figyelembe véve az adatok begépelésének számos hátrányát (Csernoch, 2017), a kontroll csoport esetében már az oktatás korai fázisaiban eltértünk a hivatalos tankönyvben írt ütemtervtől. Az adatok begépelése helyett kész táblázatokat biztosítottunk a tanulók számára és tovább léptünk a táblázatkezelői függvények használatára.

Az adatgyűjtést nyomtatott tesztek formájában végeztük minden csoportban a témakör előtt (előtesztek) és után (utótesztek). Az előtesztek kitöltésére még a témakör első órája előtt kerítettünk sort, hogy elkerüljük a hozott tudásszerkezet módosítását. Az utótesztek megírása közvetlenül az utolsó táblázatkezelés órák után történt meg. Kiemelnénk, hogy az előtesztek és az utótesztek elvégzéséhez azonos tesztlapot használtunk.

5. táblázat: A tanulók száma és a tesztek közötti eloszlás a vizsgálati és kontroll csoportokban.

	Vizsgálati csoportok		**Kontroll csoport**
	7. évfolyam	**10. évfolyam**	**7. évfolyam**
Tanulók száma	15	18	13
Előteszt	14	16	11
Utóteszt	13	13	13
Párosított tesztek	12	11	11

Az 5. táblázatban látható alacsony mintaszám azzal indokolható, hogy a közismereti informatikaoktatás csoportbontásban valósul meg, melyek létszáma általában 12–18 fő közé tehető. A csoportok számának alakulása az intézményben minden tanévben az éppen aktuális tantárgyfelosztás függvénye. A mintába tartozó alacsony csoport- és óraszám a Kerettantervben (OFI, 2013) és az iskola helyi tantervében megfogalmazott témakörök előfordulásával magyarázható. Annak ellenére, hogy minden tanuló sikeresen teljesítette a Kerettantervben meghatározott követelményeket (OFI, 2013), eltéréseket fedezhetünk fel az 5. táblázatban látható tanulói számokat illetően az elő és utóteszten résztvevők között. Ezek az eltérések tanulói hiány-

zásokkal, valamint előre nem látható iskolai tevékenységekkel magyarázhatók, melyek következtében néhány tanuló nem tudott jelen lenni a tesztek megírásakor. A párosított tesztek rekord tartalmazza azoknak a tanulóknak a számát, akik mindkét tesztelési alkalommal megjelentek.

5.1.1. Tesztlapok bemutatása

A teszt kialakításánál külön figyelmet fordítottunk arra, hogy az tartalmilag lefedje az adatkezelés és programozási aspektusait a táblázatkezelés témakörnek. További szempont volt, hogy a tanulók számukra ismeretlen adathalmazzal dolgozzanak és alkalmazzák az eddigi tanulmányaik során kialakított analitikus és számítógépes gondolkodásukat, valamint sémáikat. A tesztbe foglalt kérdések és feladatok emelkedő nehézségi szintet követnek, melyben a soron következő feladat épít az azt megelőzőbe foglalt tudáselemekre. Ennek megfelelően a teszteket két fő egységre bontottuk (Függelék 15.8):

– Alap képletkezelői és végrehajtási sorrend feladatok (*1–4.* feladatok).
– Problémák megértése, algoritmizálása, képletalkotási és függvényhasználati feladatok (*5a–g* feladatok).

Az *1.* feladatban a tanulóknak ki kellett egészíteniük egy hiányos, átlagot számító táblázatkezelői képletet. Ezzel a feladattal a célunk a tanulók táblázatkezelői függvényekkel kapcsolatos szintaktikai tudásáról való információgyűjtés volt. Ezt követte a *2.* feladat, melyben a függvények argumentumainak kezelését és formátumuk felismerését vizsgáltuk, illetve a *3.* feladat, amely a cellatartományokra való hivatkozást tesztelte. Az első egység utolsó feladatában (*4.* feladat) egy összetett tömbképlet végrehajtásának egyes lépéseit kellett a tanulóknak megfelelő sorrendben felírni. Fontos, hogy a feladat több üres sort is tartalmazott disztraktorként, mint ahány lépésből állt a képlet végrehajtása. Amennyiben a tanulók a függvények argumentumait is parancsként tüntették fel a megoldásukban, azokat az eseteket külön kezeltük és kevésbé súlyos hibának rögzítettük mintha hibás sorrendben adták volna meg a parancsokat.

A teszt második nagyobb egysége (*5a–g* feladatok) a táblázatkezelői képletek írásával megvalósított problémamegoldásra fókuszált a feladatlapba ágyazott táblázatot használva (Függelék 15.8). A feladatlapban autentikus táblázatot használtunk, melyet egy helyi cukrászda webtábláját felhasználva konvertáltunk (Mandula, 2017). A táblázat a boltban kapható torták nevét, kódjait és árait tartalmazta. Az első két feladatban a tanulóknak szét kellett választani az egy adatmezőben tárolt nevét (*5a* feladat) és kódját (*5b* feladat) a tortáknak. Ezt követte a táblázatban szövegként tárolt torták árainak konvertálása szám formátumba (*5c* feladat). A következő feladat az eldöntendő kérdések megfogalmazását vizsgálta, majd azok outputjainak megjelölését HA() függvényt használva konstans karaktersorozatokkal az IGAZ és a HAMIS ágakon (*5d* feladat). Az *5e* feladatban a tanulók feltételes megszámlálást kellett, hogy végrehajtsanak egy változó (C2) felhasználásával: azoknak a tortáknak a számára voltunk kíváncsiak, amelyek drágábbak, mint a megadott érték. Ennek és a soron következő feladatoknak a megoldását a diákok tömbképletekkel vagy probléma-specifikus függvényekkel is elvégezhették. A teszt utolsó két feladatában (*5f* és *5g* feladatok) a torták árainak összegét és átlagát kellett kiszámolni a megadott feltétel alapján: azon torták adataira voltunk kíváncsiak, melyek ára alacsonyabb, mint a C2 változóban tárolt érték (feltételes műveletvégzés).

5.1.2. Adatok feldolgozása, elemzése

Az adatok tárolására és elemzésére létrehoztunk egy adatbázist, melyben minden feladatot a bennük foglalt legkisebb tudáselemekre bontottunk. Akárcsak bármilyen programozási feladatnál, esetünkben is több helyes válasz is elfogadható volt számos feladatra, melyek elemeit ennek megfelelően alakítottuk ki. A torták árait inputként kezelő feladatoknál (*5d–5g* feladatok) a megoldást elfogadtuk attól függetlenül, hogy a tanulónak sikerült-e a konverziót elvégeznie az *5c* feladatban.

Az adatgyűjtést minden csoport esetében unplugged tesztek formájában végeztük, hogy egyértelmű és tiszta képet kapjunk a tanulók algoritmus építési gyakorlatáról és tudásáról. Továbbá ezzel a megközelítéssel elkerülhetők voltak azok a megoldások, amelyekre a diákok próbálgatások alapján találtak rá és valódi mögöttes tudást

nem tükröztek. Fontos kiemelni, hogy a 10. évfolyamos csoport tanulta és alkalmazta a lineáris keresés algoritmusát a tanórák során. Azonban mivel az alacsonyabb évfolyamok tantervében ez az egység nem szerepelt, a jelen vizsgálatból kihagyásra került.

A Sprego módszertan vizsgálatát több egységre bontottuk, hogy minél szélesebb körű és átfogóbb eredményt kapjunk:

1. A Sprego módszertan hatékonysága a témakör oktatására: a 7. évfolyamos (vizsgálati és kontroll) csoportok utótesztjeinek összehasonlítása.
2. Az előzetes, hagyományos módszerrel elsajátított tudás hatása a Sprego módszertannal szerzett tudásra: a 7. és 10. évfolyamos vizsgálati csoportok utótesztjeinek összehasonlítása.
3. A tanulók által választott problémamegoldási stratégiák aránya: az elő és utótesztekben alkalmazott függvényépítési megközelítések elemzése.

Az adatok elemzése az R szoftver segítségével történt (The R Foundation, 2019). A vizsgálat 1. és 2. egységében a csoportok közötti különbségek felfedésére és analizálására Mann–Whitney próbákat alkalmaztunk. A 3. egységben, a választott függvényalkotási módok közötti eltérések szignifikancia elemzése binomiális próbákkal történt.

5.2. Tézis vizsgálata

Követve az előző fejezetben ismertetett vizsgálati részeket, az első egységben az azonos korcsoportba tartozó tanulók (7. évfolyam) utótesztjeinek eredményeit elemeztük (vizsgálati és kontroll csoport). Az eredmények a Sprego hatékonyságát részletezik a témakör oktatására azonos előzetes tudással rendelkező és azonos korcsoportba tartozó tanulók esetén (6. táblázat).

6. táblázat: A 7. évfolyamos vizsgálati és kontroll csoportok utótesztjeinek átlagos eredményei és p-értékei feladatonkénti bontásban.

	Átlagos eredmények (%)			
Feladatok	**7. évf. vizsgálati**	**7. évf. kontroll**	**W**	**p**
1	94,87	74,36	124,5	0,0175
2	61,54	42,31	131	0,0082
3	96,15	87,18	93	0,5505
4	79,49	67,31	98	0,5009
5a	60,00	1,54	154,5	0,0001
5b	40,00	1,54	153	0,0002
5c	42,31	0,00	162,5	0,0000
5d	39,10	53,21	67,5	0,3941
5e	76,15	41,35	139,5	0,0046
5f	68,46	29,23	131,5	0,0161
5g	61,54	20,00	140	0,0041
Összesen	65,42	38,00	145	0,0013

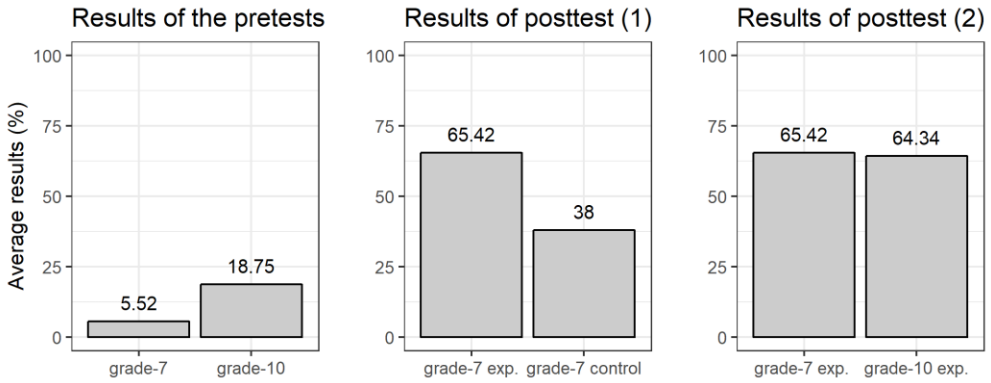
Az adatok alapján azok a tanulók, akik a Sprego módszertannal tanulták a témakört sikeresebben teljesítették a feladatokat, mint a kontroll csoport. Ez alól egyedül az *5d* feladat kivétel, amely a HA() függvény használatát vizsgálta konstans output értékekkel. Mivel korábbi kutatásainkban hasonló tendenciát figyeltünk meg, ezért fontosnak tartjuk a jelenség kihangsúlyozását. A hagyományos megközelítéssel tanuló diákok általában jobb eredményt érnek el olyan feladatokban, amelyek konstans HA() függvények alkalmazását igénylik. Ezek a tanulók azonban gyakran nem lépik túl ezt az absztrakciós szintet és nehézségeik támadnak a változó értékek használatával, valamint komplexebb algoritmusok felépítésével. Ezt a korábban megfigyelt tendenciát (Csernoch et al., 2015) alátámasztják a jelen vizsgálat *5d–5g* feladatainak statisztikai elemzései (6. táblázat). A vizsgálati csoport szignifikánsan jobb eredményeket ért el, mint a kontroll csoport az alábbi feladatokban:

– táblázatkezelői képletek szintaktikája (*1.* feladat, p = 0,0175),
– helyes argumentum formátumok (*2.* feladat, p = 0,0082),
– szöveg visszaadása konstans karakterszámmal (*5a* feladat, p = 0,0001),
– szöveg visszaadása változó karakterszámmal (*5b* feladat, p = 0,0002),
– szövegként tárolt számok számmá konvertálása (*5c* feladat, p = 0,0000),

– feltételes megszámlálás (*5e* feladat, p = 0,0046),
– feltételes összegzés (*5f* feladat, p = 0,0161),
– feltételes átlagszámítás (*5g* feladat, p = 0,0041).

Habár a vizsgálati csoport a *3.* és *4.* feladatokat is sikeresebben teljesítette, az eredmények nem mutatnak szignifikáns eltérést a két csoport között egyik esetben sem (p = 0,5505 és p = 0,5009). Továbbá, annak ellenére, hogy a kontroll csoport az *5d* feladatban magasabb pontszámot ért el, mint a vizsgálati csoport, a különbség ebben az esetben sem szignifikáns (p = 0,3941).

A teljes tesztet vizsgálva látható, hogy a vizsgálati csoport összességében is szignifikánsan jobban teljesített a kontroll csoportnál (65,42%; 38,00%; p = 0,0013) (20. ábra, középső rész).



20. ábra: A vizsgálati (grade-7/grade-7 exp. és grade-10/grade-10 exp.) és kontroll (grade-7 control) csoportok összesített százalékos eredményei az elő- és utóteszteken.

A vizsgálat második részegységében a tanulók előzetes tudásának a Sprego módszertannal kialakított tudásszerkezetre gyakorolt hatását elemeztük a 7. és 10. évfolyamos vizsgálati csoportok utótesztjei alapján. Az eredményeket a 7. táblázat tartalmazza. Azt tapasztaltuk, hogy az előzetes hagyományos és felszínes tudás hátráltatja a magas mathability problémamegoldó készségek kialakítását (Champagne et al., 1983). A 10. évfolyamos tanulók számos előnyük ellenére – kor, tapasztalat, előzetes tanórák, stb. – nem tudtak jobb eredményeket elérni, mint a 7. évfolyamos vizsgálati csoport. 11 feladatból 8 esetében (72,73%) a 7. évfolyamos csoport teljesített jobban,

azonban szignifikáns különbséget nem figyelhetünk meg. A 10. évfolyamos csoport összesen három feladatban ért el jobb eredményt:

– szöveg visszaadása konstans karakterszámmal (*5a* feladat),
– HA() függvény használata konstans outputokkal (*5d* feladat),
– feltételes átlagszámítás (*5g* feladat).

Hasonlóan az előző feladatokhoz a csoportok közötti különbség ezekben az esetekben sem szignifikáns. Kiemelnénk, hogy a 10. évfolyamos csoport eredményei a feladatok megoldásának több lehetséges megoldását is magukba foglalják akár Sprego, akár hagyományos módszert alkalmaztak a tanulók.

7. táblázat: A 7. és 10. évfolyamos vizsgálati csoportok utótesztjeinek átlagos eredményei és p-értékei feladatonkénti bontásban.

	Átlagos eredmények (%)			
Feladatok	**7. évf. vizsgálati**	**10. évf. vizsgálati**	**W**	**p**
1	94,87	92,31	78	0,6524
2	61,54	59,62	88	0,8691
3	96,15	93,59	72,5	0,4229
4	79,49	72,44	73	0,5654
5a	60,00	69,23	91,5	0,7028
5b	40,00	39,23	81	0,8762
5c	42,31	34,62	73,5	0,5831
5d	39,10	57,69	111,5	0,1708
5e	76,15	59,04	57,5	0,1666
5f	68,46	66,92	75	0,6380
5g	61,54	63,08	86	0,9586
Összesen	65,42	64,34	77,5	0,7388

A teljes tesztet egybefogva láthatjuk, hogy a 10. évfolyamos tanulók eredményei (64,34%) összesítve sem jobbak, mint a 7. évfolyamos diákoké (65,42%), illetve, hogy szignifikáns különbség továbbra sem figyelhető meg ($p = 0,7388$) (20. ábra, jobb oldal).

Azon túlmenően, hogy a 7. táblázatban látható eredmények alapján nincs szignifikáns különbség a két Sprego csoport eredményei között, fontos ismét kiemelnünk, hogy a 10. évfolyamos csoport korábban már két évfolyamban is tanulta a témakört.

Továbbá, hogy korosztályukból adódóan magasabb kognitív és absztrakciós készségekkel rendelkeznek. Ennek ellenére az azonos tesztek kitöltése minimális eltérést eredményezett a csoportok között. Azon túlmenően, hogy a korábbi hagyományos megközelítések hogyan befolyásolják a 10. évfolyamos tanulók problémamegoldó készségeit, fontos az eredményeket is szem előtt tartanunk, melyek bizonyítják, hogy a felületi megközelítések nem támogatják a szekvenciális, térbeli tájékozódás és emlékezés, memorizálás készségek kialakulását (Kruck et al., 2003).

A vizsgálat harmadik egysége a tanulók által választott függvényépítési megközelítések elemzésére irányult. Más megfogalmazásban, hogy az egyes csoportok tanulói Sprego vagy probléma-specifikus függvényeket alkalmazva oldották meg a feladatokat. A 10. évfolyamos csoport esetében feltételeztük, hogy a tanulókat befolyásolja a Sprego módszertan, azonban korábbi eredmények alapján annak a lehetőségét is szem előtt tartottuk, hogy az előzetes tudásuk negatív hatással lesz az előrehaladásukra (Champagne et al., 1983; Kruck et al., 2003; Moyo et al., 2016). Az elemzés ezen részéhez a tesztek *5e–5g* feladatait vizsgáltuk (8. táblázat), mivel ezeknek a feladatoknak a lehetséges megoldási módjai mutatnak jelentős eltéréseket az alkalmazott módszertől függően.

8. táblázat: Az *5e–5g* feladatok Sprego megoldásai. A képletek szemléltetik, hogy a megoldások azonos algoritmuson alapulnak (4.2.1 fejezet).

Feladat	Megoldás
5e	{=SZUM(HA(B2:B58>C2;1))}
5f	{=SZUM(HA(B2:B58<C2;B2:B58))}
5g	{=ÁTLAG(HA(B2:B58<C2;B2:B58))}

A feladatokhoz kapcsolódó tanulói számokat a 9. táblázat tartalmazza csoport és alkalmazott módszer szerinti bontásban. Mivel a tanulók nem rendelkezhettek előzetes tudással a Sprego módszertanról mielőtt megkezdődött a témakör oktatása, az üres kitöltéseket is a hagyományos módszer kategóriájába soroltuk. A táblázat adatai alapján látható, hogy minimális eltéréseket leszámítva mindkét 7. évfolyamos csoport azt a megközelítést követi, amelyet a tanórákon alkalmaztak. Ezzel szemben jelentősebb eltéréseket figyelhetünk meg a 10. évfolyamos utóteszteket illetően.

9. táblázat: A hagyományos (Hagy.) és Sprego megközelítéseket alkalmazott tanulók száma tesztek, vizsgált feladatok és csoportok szerinti bontásban. Az üres kitöltések a hagyományos módszerhez számolva.

		7. évf. vizsgálati		**7. évf. kontroll**		**10. évf. vizsgálati**	
Teszt	**Feladat**	**Hagy.**	**Sprego**	**Hagy.**	**Sprego**	**Hagy.**	**Sprego**
Előteszt	5e	14	0	11	0	14	2
	5f	14	0	11	0	13	3
	5g	14	0	11	0	12	4
Utóteszt	5e	1	12	13	0	5	8
	5f	1	12	13	0	2	11
	5g	1	12	12	1	3	10

A 10. évfolyamos vizsgálati csoport közel minden tagja probléma-specifikus függvényeket használva oldotta meg a feladatokat az előtesztben. Az utótesztben azonban ez az arány megváltozott: több tanuló is a Sprego megoldásokat részesítette előnyben. A feltételes megszámlálás – *5e* feladat – mutatja a legkisebb eltérést: mindössze három fővel választották többen a Sprego megoldást, mint a tradicionálist. Ez a jelenség az elterjedt DARABTELI() függvénynek tulajdonítható, amelyet a tanulók informatikai ismereteik korai szakaszaiban sajátítanak el, illetve ismételnek valahányszor újra találkoznak a témakörrel. Ezzel szemben a feltételes összegzés és átlagszámítás (*5f* és *5g*) feladatokat a tanulók Sprego függvényekkel oldották meg (8. táblázat) minimális számú kivétellel (kettő és három fő). Ez a tendencia, miszerint a Sprego módszertannal tanult csoportok annak megoldási megközelítésére váltanak, jól megfigyelhető a 9. táblázat alapján. Azonban a 10. évfolyamos utóteszteket elemezve a feltételes megszámlálás (*5e* feladat) és átlagszámítás (*5g* feladat) esetében a tanulók nem mutattak szignifikáns eltérést a választott módszerek tekintetében ($p = 0{,}5811$ és $p = 0{,}0923$). Ezzel szemben az *5f* feladat esetében, amely megoldása ugyanazt az algoritmust követi, mint az *5e* és *5g* feladatok, a különbség szignifikánsnak bizonyult ($p = 0{,}0225$).

A 7. és 10. évfolyamos vizsgálati csoportok problémamegoldó megközelítéseinek összevetése egyértelműen mutatja, hogy a tanulók korábbi tudása befolyásolja döntéseiket a függvények kiválasztásában. Ezt az állítást a tesztek javítása során szerzett tapasztalataink is igazolják. Több olyan esettel is találkoztunk, ahol 10. évfolyamos

tanulók mindkét módszert próbálták sikertelenül alkalmazni, vagy olyan probléma-specifikus megközelítéssel akarták megoldani a feladatot, melynek alkalmazási módjára nem emlékeztek. Találkoztunk HA() függvénnyel, amelyet a DARAB-TELI() függvény logikájával és argumentumaival próbáltak használni, továbbá SZUM() és ÁTLAG() függvényekkel, melyek zárójeleibe a SZUMHA() és ÁT-LAGHA() függvények argumentumai lettek megadva.

A 10. táblázatban látható a tanulók száma és az általuk választott megoldási megközelítés az üres kitöltések figyelmen kívül hagyásával. A hagyományos stratégiát választó tanulók számát vizsgálva jelentős esést figyelhetünk meg a 9. táblázat adataihoz képest. Ez alól egyedüli kivétel a 7. évfolyamos kontroll csoport utótesztje. A 10. évfolyamos utótesztek adatait elemezve a korábban megfigyelt mintát láthatjuk ismét megjelenni, miszerint a tanulók többnyire a Sprego módszertannal történő feladatmegoldást preferálják. A választott módszerek közötti szignifikancia vizsgálat némileg eltér az előző elemzés eredményeitől: $p = 0{,}2266$ (*5e* feladat), $p = 0{,}0064$ (*5f* feladat) és $p = 0{,}0117$ (*5g* feladat). Ebből következik, hogy amennyiben az üres kitöltéseket figyelmen kívül hagyjuk, a tanulók szignifikáns különbséggel választják a Sprego módszertant a feladatok megoldására a három feladatból két esetben (66,67%). Érdekességként kiemelnénk, hogy a 10. évfolyamos tanulók előtesztjei alapján a tanulók többsége – 81%, 81% és 75% a tárgyalt feladatok sorrendjében – az előzetes tudásukra támaszkodva hozzá sem tudott kezdeni a feladatok megoldásához.

Az eredmények alátámasztják, hogy a hagyományos felületi alapú módszerek és a tapasztalatlapú tudás (naive knowledge) negatív hatással vannak a tanulók problémamegoldó készségeire.

10. táblázat: A hagyományos (Hagy.) és Sprego megközelítéseket alkalmazott tanulók száma tesztek, vizsgált feladatok és csoportok szerinti bontásban. Az üres kitöltések figyelmen kívül hagyva.

		7. évf. vizsgálati		**7. évf. kontroll**		**10. évf. vizsgálati**	
Teszt	**Feladat**	**Hagy.**	**Sprego**	**Hagy.**	**Sprego**	**Hagy.**	**Sprego**
Előteszt	5e	0	0	0	0	1	2
	5f	0	0	0	0	0	3
	5g	0	0	0	0	0	4
Utóteszt	5e	0	12	10	0	3	8
	5f	0	12	10	0	1	11
	5g	0	12	8	1	1	10

A bemutatott három vizsgálati egység eredményei alapján a Sprego módszertan azonos korcsoportban lévő és megegyező előismeretekkel rendelkező tanulók oktatásában szignifikánsan hatékonyabb, mint a hagyományos megközelítések. Továbbá, a korábban megszerzett, tradicionális módszerekkel elsajátított tudás negatívan befolyásolja a Sprego módszertannal történő előrelépést és készségfejlesztést. Azok a tanulók, akik hagyományos és Sprego megközelítésekkel is tanulták a táblázatkezelés témakört, inkább választják a Sprego módszertan algoritmusépítését és általános célú függvénybázisát a feladatok megoldására, mint a probléma-specifikus függvényeket. Az eredmények alapján a [T1] tézis, miszerint a Spreadsheet Lego módszertan szignifikánsan hatékonyabb a táblázatkezelés oktatására, mint a hagyományos, felületorientált megközelítések, igazolt (Angeli, 2013; Baranyi & Gilanyi, 2013; Csapó et al., 2019; Csernoch, 2014; Csernoch & Biró, 2014b, 2017; Garrett, 2015).

6. [T2] Spreadsheet Legoval hosszú távú tudás kialakítása

6.1. Alkalmazott módszerek, eszközök és eljárások

A tézisbe foglalt vizsgálat során összehasonlítottuk egy középiskola diákjaiból álló mintán késleltetett utótesztek segítségével végzett mérésnek az eredményeit a Sprego és a hagyományos megközelítésekkel kialakított hosszú távú tudás hatékonyságának elemzésére. A mintát egy helyi középiskola hét csoportja alkotta. A csoportok kiválasztásánál kiemelt szempont volt, hogy a tanulók a vizsgálatot megelőző tanévben tanulták a táblázatkezelés témakört akár hagyományos, akár Sprego módszert használva. A tanulók korosztályát tekintve két 8. évfolyamos és két 10. évfolyamos Sprego módszertannal tanult (vizsgálati csoportok) és három 10. évfolyamos tradicionális módszereket használó csoportot (kontroll csoportok) vizsgáltunk. A két 8. évfolyamos vizsgálati csoport, valamint a kontroll csoportok közül két csoport hat évfolyamos képzésben vett részt. A táblázatkezelés témakört egy évfolyammal korábban tanulta minden mintában szereplő tanuló a Kerettanterv (OFI, 2013) és az iskola helyi tanterve alapján. Ennek megfelelően a vizsgálati csoportok közül a 8. évfolyamos csoportok előző tanévben egy óra/hét, 8 héten keresztül, a 10. évfolyamos csoportok pedig egy óra/hét, 6 héten keresztül beosztással tanulták a témakört. A kontroll csoportok esetében az óraszámokat befolyásolta a két hat évfolyamos képzésben résztvevő csoport: két 10. évfolyamos csoport két óra/hét, 6 héten keresztül, illetve azt megelőzően 7. évfolyamban egy óra/hét, 8 héten keresztül. A kontroll csoportok további 10. évfolyamos csoportja pedig egy óra/hét, 6 héten keresztül beosztással sajátította el a témakört.

Az adatgyűjtést nyomatott formában, késleltetett utótesztek segítségével végeztük. A tesztek kitöltésére a tanév utolsó szakaszában, május-június hónapokban került sor a csoportok időbeosztásától függően.

11. táblázat: A késleltetett utótesztet kitöltött tanulók száma a vizsgálati és kontroll csoportokban.

	Vizsgálati csoportok				**Kontroll csoportok**		
	8(1)	8(2)	10(1)	10(2)	10(1)	10(2)	10(3)
Kitöltések száma	18	18	17	15	7	13	7

A vizsgálatban résztvevő teljes minta nagysága N = 95. A tanulók eloszlását a vizsgálati (N = 68) és kontroll (N = 27) csoportok között a 11. táblázat részletezi. A kontroll csoportok kisebb mérete a vizsgálati csoportok méretéhez képest a tanév végi időbeosztással indokolható. A csoportok oktatását a kutatócsoportunkon kívül álló informatika tanárok végezték, melynek következményeként a tanulói hiányzásokból és a tanév végére időzített iskolai programokból eredő mulasztásokat nem volt lehetőségünk pótolni. További befolyásoló tényezőkként voltak jelen a csoportok méretére és számára vonatkozóan az 5.1 fejezetben ismertetett tantárgyspecifikus sajátosságok.

6.1.1. Tesztlapok bemutatása

A tanulók tudásának mérésére a TAaAS felmérésben is használt táblázatkezelői kérdéseket alkalmaztuk (Függelék 15.9). A feladatlap a tanulók képletalkotási és függvényhasználati képességeit vizsgálta öt feladaton keresztül (*a–e* feladatok), illetve a képletelemzési tudásukat tesztelte az utolsó feladatban (*f* feladat). A feladatok kidolgozásához a tanulóknak a mellékelt, világ országainak adatait tartalmazó táblázat kivonatát kellett használniuk (OFI, 2004). A táblázat az országok nevét, földrészét, fővárosát, területét és lakosságát (ezer főben megadva) tartalmazta egy kijelölt változóval együtt (G2).

Az első feladatban (*a* feladat) a tanulóknak a lineáris keresés algoritmusát felhasználva kellett megadniuk a legnagyobb területű ország fővárosát. Ezt követte egy népsűrűséget kiszámító képlet írása (*b* feladat). Ennek a feladatnak a teljesítéséhez a tanulóknak a táblázatkezelői ismereteken kívül tisztában kellett lenniük a népsűrűség kiszámításának módjával, valamint figyelembe kellett venniük, hogy a lakosság értékek 1000 főben értendők. A *c* feladatban feltételes megszámlálást kellett végezni

egy konstans értéket felhasználva, melyet egy feltételes átlagszámítás követett bevezetve a G2 változót a feladatmegoldásba (*d* feladat). A következő probléma ismételten egy feltételes megszámlálást igénylő feladat volt (*e* feladat), azonban ezúttal más inputra és feltételre támaszkodva. Az utolsó feladatban a tanulóknak egy összetett tömbképletet kellett elemezniük (*f* feladat), majd természetes nyelven megfogalmazniuk annak funkcióját.

6.1.2. Adatok feldolgozása, elemzése

A tesztek eredményeit ezúttal is egy saját tervezésű adatbázisban tároltuk, melyben minden feladatot a bennük foglalt tudáselemek számával azonos elemekre bontottunk. Mivel a *c*–*e* feladatok megoldhatók voltak a Sprego módszertanban alkalmazott tömbképletek, illetve a táblázatkezelői környezetbe beépített probléma-specifikus függvények segítségével is, ezeket az eseteket külön kezeltük.

A javítás során amennyiben a tanuló az adott függvény angol megnevezését használta (például SZUM() helyett SUM()), a választ helyesnek tekintettük. Továbbá, ha a válaszban szereplő argumentumok sorrendje megfelelő volt, azonban a zárójelezés során a tanuló téves helyen zárta le a függvényt, a választ ismételten elfogadtuk. Fontosnak tartjuk kiemelni, hogy az *a* feladatba foglalt keresőalgoritmust a 8. évfolyamos tanulók nem tanulták, azonban a felsőbb évfolyamok esetében indokolt volt a feladatlapon való szerepeltetése.

Az adatok elemzését ismételten az R szoftverrel végeztük (The R Foundation, 2019). Az adatok normalitásának vizsgálata Shapiro-Wilk féle normalitás teszt segítségével történt, illetve a csoportok közötti eltérések szignifikanciájának vizsgálatára Mann–Whitney próbákat alkalmaztunk.

6.2. Tézis vizsgálata

A késleltetett utóteszten elért eredményeket tanulói csoportok szerinti bontásban a 12. táblázat, vizsgálati és kontroll csoportok szerinti összesítésben pedig a 13. táblázat tartalmazza. A tesztlapok bemutatásánál kiemeltük, hogy a *c*–*e* feladatok meg-

oldása két lehetséges utat követhet: a Sprego módszertanban is alkalmazott tömbképletek építését, illetve a táblázatkezelői környezetbe beépített probléma-specifikus függvények használatát. Az adatok elemzése során a feladatok teljesítettségének tekintetében a két megoldási stratégiát önállóan kezeltük minden tanuló esetében. A csoportok által (tanulói és vizsgálati típus szerint) elért feladatonkénti eredmények összesítésében azonban a két megoldási módszerrel szerzett pontszámokat egybefogtuk.

12. táblázat: A vizsgálati és kontroll csoportok osztályonként elért százalékos eredményei a késleltetett utóteszten feladatonkénti bontásban.

	Vizsgálati csoportok (%)				**Kontroll csoportok (%)**		
Feladat	**8(1)**	**8(2)**	**10(1)**	**10(2)**	**10(1)**	**10(2)**	**10(3)**
a	5,19	6,67	12,94	18,22	3,81	3,08	0,00
b	20,83	19,44	50,00	23,33	14,29	32,69	0,00
c	65,00	73,89	62,94	67,33	21,43	38,72	11,43
d	35,19	59,26	64,71	61,48	25,40	19,74	0,00
e	48,77	59,26	65,36	56,30	7,94	20,62	7,94
f	57,41	77,78	74,51	75,56	14,29	56,41	23,81
Összesen	38,73	49,38	55,08	50,37	14,52	28,54	7,20

Az eredmények alapján láthatjuk, hogy a vizsgálati csoportok minden feladat esetében jobb eredményeket értek el. Ez alól egyedül a 8(2) vizsgálati csoport *b* feladatban elért értéke kivétel, melynél a 10(2) kontroll csoport magasabb százalékot szerzett. Fontosnak tartjuk megjegyezni, hogy a kontroll csoportok teljesítményéhez nagy mértékben hozzájárul a 10(3) kontroll csoport alacsony eredménye, mely alapján három feladatban is 0 pontot szereztek a kitöltők, illetve további két feladat esetén 12,00% alatti pontszámokat értek el (12. táblázat). A tanulói csoportok összesített százalékos eredményeit figyelembe véve a legjobban a 10(1) vizsgálati csoport (55,08%), leggyengébben pedig a 10(3) kontroll csoport (7,20%) teljesített. Érdekesség, hogy a 8(2) vizsgálati csoport összteljesítménye közel azonos a 10(1) és 10(2) vizsgálati csoportok százalékaival annak ellenére, hogy a tanulók között két évfolyamnyi korkülönbség van. Ugyancsak fontos kiemelni, hogy a kontroll csoportok a

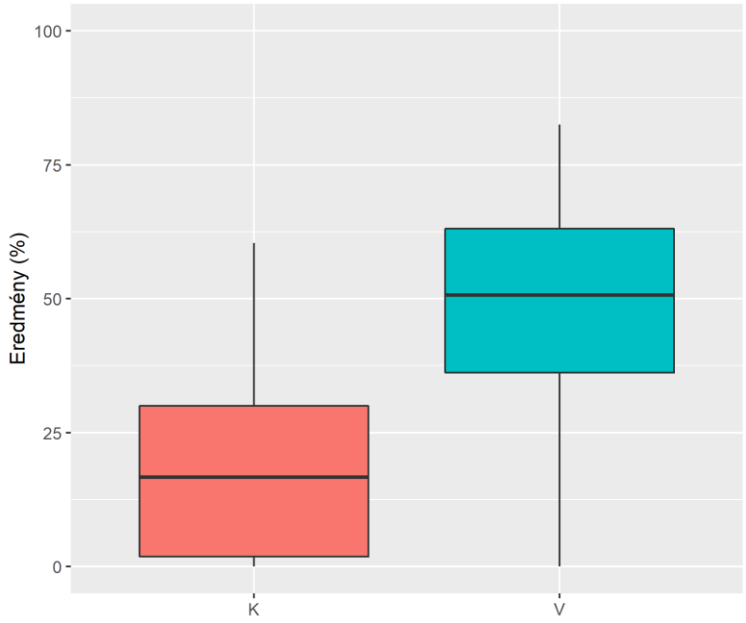
szintén két évfolyamnyi korkülönbség ellenére – az említett *b* feladat esetét leszámítva – egyik feladatban sem tudtak olyan magas pontszámot elérni, mint a Sprego módszertannal tanult 8. évfolyamos tanulók.

13. táblázat: A vizsgálati és kontroll csoportok összesített eredményei a késletetett utóteszten feladatonkénti bontásban a csoportok közötti eltéréseket mutató p-értékekkel.

Feladat	**Vizsgálati csoportok (%)**	**Kontroll csoportok (%)**	**W**	**p**
a	10,39	2,47	404	0,0000
b	28,31	19,44	720	0,0841
c	67,35	27,16	366	0,0000
d	54,74	16,09	302,5	0,0000
e	57,35	14,04	302,5	0,0000
f	71,08	37,04	517	0,0005
Összesen	48,20	19,37	264	0,0000

A Sprego módszertan algoritmusépítési és sémakialakítási gyakorlatának hatékonyságát jól szemléltetik a *c-e* feladatokra kapott eredmények. A vizsgálati csoport tanulóinak százalékos értékeiben ugrás tapasztalható ezeknél a feladatoknál. Ezzel szemben a kontroll csoport tanulói nehézségekbe ütköztek a probléma-specifikus függvények alkalmazásakor. A tesztek javítása során számos alkalommal találkoztunk függvény töredékekkel, illetve olyan függvényfelépítésekkel, amelyek nem követték a táblázatkezelői környezet funkcionális nyelvének szabályait. Ezek a tapasztalatok rávilágítottak, hogy a kontroll csoport tanulói több esetben sem tudták a feladat megoldásához szükséges megfelelő eszközöket kiválasztani.

A képletelemzés (*f* feladat) során három Sprego csoport is 74,00% felett teljesített, illetve egy csoport 57,41%-os eredményt ért el. Ezzel szemben a kontroll csoportok közül a legjobban teljesítő 10(2) csoport 56,41%-os értéke sem éri el a legrosszabbul teljesítő vizsgálati csoport szintjét, valamint a 10(1) és 10(3) kontroll csoportok a 15,00%-os és 25,00%-os eredményeket sem. A csoportok összesített eredményei közötti eltérést szemlélteti a 21. ábra.



21. ábra: A kontroll (K) és vizsgálati (V) csoportok százalékos eredményeinek eloszlása a késletetett utóteszt eredményei alapján.

A 13. táblázatba foglalt adatok alapján a vizsgálati csoportok szignifikánsan jobban teljesítettek a kontroll csoportoknál. Ez alól a már korábban említett *b* feladat kivétel, ahol a Mann-Whitney féle próba nem mutatott ki szignifikáns eltérést ($p = 0{,}0841$). Ez azzal magyarázható, hogy a feladat táblázatkezelői függvények használata helyett egy, a népsűrűséget kiszámító képlet írását kérte. A tesztekben megadott válaszok alapján a tanulók többsége (a vizsgálati és a kontroll csoportokban egyaránt) nem a képletíráshoz kapcsolódó tudáselemek hiánya miatt ért el alacsony pontszámot, hanem mert nem ismerték a népsűrűség kiszámításának matematikai módját.

A jelen tézishez kapcsolódó további eredményeket szolgáltat a [T1] tézisnél végzett vizsgálat előtesztjeinek elemzése. A 7. évfolyamos vizsgálati és kontroll csoportok előtesztjeit egybefogva az eredményeket összehasonlítottuk a 10. évfolyamos tanulók előtesztjeivel. Azt vizsgáltuk, hogy az előzetes tudással nem rendelkező 7. évfolyamos tanulókhoz képest a témakört már több évfolyamon keresztül hagyományos módszerrel tanuló 10. évfolyamos diákok milyen szintű tudással rendelkeznek. A vizsgálat eredményeit a 14. táblázat tartalmazza.

14. táblázat: A 7. (vizsgálati és kontroll csoportok összesítve) és 10. évfolyamos csoportok előteszt-jeinek átlagos eredményei és p-értékei feladatonkénti bontásban.

	Átlagos eredmények (%)			
Feladatok	**7. évfolyam**	**10. évfolyam**	**W**	**p**
1	20,00	64,58	356,5	0,0000
2	25,00	25,00	220	0,5829
3	2,67	44,79	329	0,0000
4	13,00	49,48	330	0,0002
5a	0,00	2,50	225	0,0794
5b	0,00	0,63	212,5	0,2301
5c	0,00	0,00	200	
5d	0,00	7,81	262,5	0,0036
5e	0,00	3,91	237,5	0,0285
5f	0,00	3,75	237,5	0,0285
5g	0,00	3,75	250	0,0102
Összesen	5,52	18,75	380	0,0000

Az adatok alapján látható, hogy a 7. évfolyamos tanulók minimális előzetes tudással oldották meg a feladatokat, mivel a témakört korábbi informatikai tanulmányaik során nem tanulták. Ebből következik, hogy a 10. évfolyamos tanulók négy feladatot leszámítva minden esetben szignifikánsan jobb eredményeket értek el. A *2.* feladatban mindkét évfolyam 25,00%-ot ért el szignifikáns különbség nélkül ($p = 0,5829$). Ez szemlélteti, hogy a 10. évfolyamos tanulók nem ismerték fel a helyes argumentum formátumokat nagyobb arányban, mint azok a diákok, akik még nem dolgoztak táblázatkezelői függvényekkel korábban. Hasonló tendencia figyelhető meg a szövegfüggvények használatát igénylő *5a–5c* feladatokban, ahol annak ellenére, hogy a 10. évfolyamos csoport jobb eredményeket ért el, szintén nem mutatható ki szignifikáns eltérés ($p = 0,0794$, $p = 0,2301$). Kiemelnénk, hogy az *5c* feladatot mindkét csoport 0,00 ponttal teljesítette, ezért a szignifikanciavizsgálat nem volt elvégezhető Mann-Whitney próbával. Ez a jelenség arra vezethető vissza, hogy a szövegfüggvények oktatása nem része a tradicionális táblázatkezelés oktatásnak annak ellenére, hogy a programozásban jelentős szerepük van.

A teljes teszt eredményeit vizsgálva a 7. évfolyamos csoportok 5,52%-ot értek el, míg a 10. évfolyamos csoport 18,75%-os eredménnyel teljesítette azt, a két évfolyam

közötti szignifikáns különbséggel (p = 0,0000) (20. ábra, bal oldal). Fontos azonban kiemelni, hogy a 10. évfolyamos tanulók ezt a szignifikáns különbséget nagyon alacsony pontszámokkal (8% alatt) érték el a teszt második felében (*5a–5g* feladatok, képletalkotás és függvények használata) szemben a 7. évfolyam 0,00%-os értékeivel. A 10. évfolyam korábban ismertetett előnyeit figyelembe véve a magasabb absztrakciós és kognitív készségeik az elért pontszámoknál magasabb százalékos értékeket indokoltak volna.

Az ismertetett eredmények alapján a késleltetett utóteszteket kitöltő csoportok közül azok a tanulók, akik korábban Sprego módszertannal tanultak, szignifikánsan jobb eredményeket értek el, mint a hagyományos megközelítést alkalmazó csoportok. Ezen túlmenően a táblázatkezelés témakör kezdetén kitöltött előteszek alapján a több évfolyamos tapasztalattal rendelkező 10. évfolyamos diákok a szignifikáns különbség ellenére minimális százalékkal értek el magasabb eredményeket, mint azok a diákok, akik korábban nem tanulták a témakört. A fentiekből következően a [T2] tézis, mely szerint a séma-alapú algoritmusépítésre alapozott táblázatkezelői módszertan oktatása hosszú távú tudást alakít ki, igazolt (Biró & Csernoch, 2015b; Csernoch, 2016, 2017, 2019b). Az eredmények publikálása a közeljövőben várható.

7. [T3] Dokumentumelemzés és hibafelismerés

7.1. Alkalmazott módszerek, eszközök és eljárások

A tézis elemzése során arra a kérdésre kerestük a választ, hogy az elsőéves hallgatók a hozott tudásszerkezetük alapján rendelkeznek-e a jól formázott dokumentumok elkészítéséhez nélkülözhetetlen készségekkel: a dokumentumok azonos szerkezeti egységei közötti összefüggések meglátásához szükséges algoritmikus készséggel és számítógépes gondolkodással. Ezen készségek vizsgálatára bekapcsolódtunk a 2011-ben indult TAaAS projektbe (Csernoch et al., 2015). A projekt keretein belül az informatikus képzésre felvételt nyert elsőéves hallgatók algoritmikus és alkalmazói tudásának tesztelése történt meg, minden évben az első tanulmányi hét folyamán. Az időzítésben fontos szempont volt, hogy a hallgatók még azelőtt számot adjanak tudásukról, mielőtt felsőfokú tanulmányaik során új ismeretanyagokat sajátítanának el. A TAaAS az informatika számos aspektusát vizsgálta: a készségek szintjét önértékelés alapján, az alapvető számítógép használati ismereteket, a szakterminológia használatát, valamint az algoritmikus készséget és számítógépes gondolkodást szöveg-, táblázatkezelői és programozási környezetekben. Munkánk során a felmérés szövegkezelői környezetekre fókuszáló egységével dolgoztunk.

A Debreceni Egyetem Informatikai Karára 2014-ben felvételt nyert Gazdaságinformatikus BSc (GI) és Programtervező informatikus BSc (PTI) hallgatók eredményeit vizsgáltuk. A mintában (N = 203) szereplő hallgatók létszámának eloszlását és informatika érettségi eredményeit a 15. táblázat tartalmazza.

15. táblázat: A vizsgálatban résztvevő GI és PTI csoportokba tartozó hallgatók száma, eloszlása és informatika érettségi eredményei.

		Érettségik száma (N)			**Érettségi eredmények (%)**	
	N	**Közép**	**Emelt**	**Nincs adat**	**Közép**	**Emelt**
GI	86	58	6	24	74,00	53,67
PTI	117	84	47	2	82,00	67,13

A hallgatók többsége közvetlenül a középiskolai tanulmányaik befejezése után nyert felvételt az egyetemre, amely a vizsgálat alapjául szolgáló hozott tudás aktualitását igazolja. Az informatika érettségi vizsgák számának tekintetében középszinten a GI hallgatók 67,44%-a, a PTI hallgatók 71,79%-a szerzett bizonyítványt. Ez az arány az emelt szintű informatika érettségiket vizsgálva a következőképp alakul: GI: 6,98% és PTI: 40,17%. Fontosnak tartjuk kiemelni, hogy 26 fő nem adott meg adatot az általa teljesített érettségi vizsgáról, valamint, hogy több hallgató is rendelkezik egyszerre közép- és emelt szintű informatika érettségivel is. A 15. táblázatban látható érettségi adatok alapján feltételezhetjük, hogy valamennyi kitöltő megalapozott szövegkezelői ismeretek birtokában van, illetve, hogy a PTI csoport tagjai jobb eredményeket fognak elérni a teszten.

7.1.1. Tesztlapok bemutatása

A TAaAS feladatsor (Csernoch & Biró, 2015b; Csernoch, 2019a) szövegkezelést vizsgáló részénél a hallgatóknak egy idegen nyelvű dokumentum számozott elemeit kellett hibaként felismerniük, típus szerint beazonosítaniuk, majd a dokumentumkivonat alatt feltüntetett táblázatban megjelölniük a megfelelő hibakategóriát (Függelék 15.10): tördelési, formázási, szintaktikai, szemantikai és tipográfiai (Csernoch, 2009). A dokumentum két ponton is segítséget nyújtott a hallgatóknak a hibák beazonosításában:

- A kivonaton látszódtak a nem nyomtatható karakterek, illetve a beszúrt kép körül megjelenő transzformációs jelölők a kép horgonyzási pontjával. Ez számos hiba azonosításához nélkülözhetetlen információ, azonban a nyomtatásban látható hibák felismerését is elősegíti.
- A kivonat egyes hibák megjelölését nagyítással, vagy külön kiemeléssel pontosította.
- A szöveg idegen nyelven íródott. Erre támaszkodva a hallgatók feltételezhették, hogy a magyar nyelvű teszt nem várja el tőlük az idegen nyelvű szöveg tartalmi megértését és kizárhatták – amennyiben tudták a hibakategória jelentését – a szemantikai hibakategóriát.

7.1.2. Adatok feldolgozása, elemzése

A tesztek kijavítása után nyert adatok tárolására a korábbiakban ismertetett gyakorlatot követve egy saját tervezésű adatbázist készítettünk. A feladatlap sajátosságaiból eredően minimális elemszámmal dolgoztunk.

A vizsgálati adatok elemzése során nem feltételeztük, hogy minden kitöltő pontosan ismeri a szakterminológiát, ezért a megfelelő hibakategóriák megjelölése helyett arra helyeztük a hangsúlyt, hogy a hallgatók meglátták-e az összefüggéseket az azonos hiba-előfordulások között. Más megfogalmazásban: Rendelkeztek-e a szükséges szintű algoritmikus készséggel és számítógépes gondolkodással ahhoz, hogy felismerjék a kapcsolatokat egy dokumentum azonos típusú szerkezeti egységei között? Ennek megfelelően a teszt *2*-es és *11*-es pontjait (üres bekezdések használata a térköz bekezdésformázás helyett), valamint *6*-os és *7*-es pontjait (bekezdésen belül igazítás szóközök használatával) párosítottuk. Mindkét pár hibái a tördelési hibakategóriába sorolhatók, azonban jelen vizsgálatban a közöttük lévő kapcsolat felfedezésének tekintettük, ha a pár tagjait azonos (akár téves) hibakategóriába sorolták a hallgatók. A vizsgálatot két egységre bontottuk a főbb elemzési lépések elkülönítése céljából:

1. Az azonos hibakategóriába tartozó hibaesetek közötti kapcsolatok felismerése.
2. A teszten elért eredmények és a középszintű informatika érettségi eredmények összevetése.

Az adatelemzés részét képezte az adatok több szempont szerinti vizsgálata: megvizsgáltuk a hibapárok közötti összefüggések meglátását, illetve az üresen hagyott kitöltéseket is számba vettük. Az összefüggések összesítését követően külön kategóriába különítettük el azokat az eseteket, amelyekben csak az egyik, illetve amelyekben mindkét pár közötti összefüggést felismerte a hallgató. Következő lépésként elvégeztük az informatika érettségi eredmények vizsgálatát a csoportokban. A közép- és emelt szintű informatika érettségi eredmények és a hibafelismerés során kapott adatok közötti összefüggéseket is elemeztük mindkét minta esetében. Az adatok

elemzése táblázatkezelői környezetben történt, a Microsoft Excel szoftvert használva (Microsoft, 2019b).

7.2. Tézis vizsgálata

Az elsőéves informatika szakos hallgatók szövegkezelési ismereteinek vizsgálatát az előző fejezetben ismertetett két egységben tárgyaljuk. Az első egységben az azonos hibakategóriába tartozó hibaesetek közötti kapcsolatok felismerésének arányát elemeztük. A teszek eredményeit a 16. táblázat tartalmazza. A táblázatban külön választottuk és csoportosítottuk az eredményeket az alábbiak szerint:

– A hallgató azonos hibatípust jelölt meg a hibapár mindkét tagjánál (felismerte a köztük lévő kapcsolatot) függetlenül a jelölt hibakategória helyességétől (2. és 3. oszlopok).
– A hallgató által megjelölt hibakategóriák azonosak és helyesek is (tördelési hiba) (4. és 5. oszlopok).
– A hallgató eltérő hibakategóriákat jelölt meg a hibapárok tagjainak esetén, avagy nem ismerte fel a közöttük lévő kapcsolatot (6. és 7. oszlopok).
– A hallgató a hibapár mindkét tagját üresen hagyta, azokat nem sorolta be egyik kategóriába sem (8. és 9. oszlopok). Ezek az értékek azzal magyarázhatók, hogy a hallgató a feladatlap kitöltésekor nem foglalkozott az érintett részfeladattal, vagy pedig képtelen volt azt bármelyik kategóriába is besorolni az előzetes tudása alapján.

A *2–11* feladatpárban (üres bekezdések használata térköz helyett) lévő kapcsolatokat a 16. táblázat adatai alapján a hallgatók kevesebb, mint fele azonosította. Ez az arány tovább csökken, amennyiben a helyes megjelöléseket vizsgáljuk. A PTI csoportban lévő kitöltők érték el a legmagasabb pontszámot, mely alapján közel a hallgatók harmada ismerte fel az üres bekezdéseket tördelési hibaként. A *6–7* hibapárt (igazítás szóközökkel) vizsgálva a kapcsolatot – a két csoport eredményeit összesítve – kevesebb, mint a hallgatók harmada ismerte fel. Emellett a helyes hibakategória megjelölése kevesebb, mint a kitöltők 3%-ának sikerült.

16. táblázat: A *2–11* és *6–7* hibapárokban felismert kapcsolatok a Gazdaságinformatikus (GI) és Programtervező informatikus (PTI) csoportokban.

	Azonos hibatípus		**Helyes**		**Eltérő hibatípus**		**Mindkettő üres**	
	2–11	**6–7**	**2–11**	**6–7**	**2–11**	**6–7**	**2–11**	**6–7**
GI (N)	36	26	12	2	26	24	10	23
GI (%)	41,86	30,23	13,95	2,33	30,23	27,91	11,63	26,74
PTI (N)	57	32	39	3	24	36	15	32
PTI (%)	48,72	27,35	33,33	2,56	20,51	30,77	12,82	27,35
Összesen (N)	93	58	51	5	50	60	25	55
Összesen (%)	45,81	28,57	25,12	2,46	24,63	29,56	12,32	27,09

A két hibakategória eredményeinek összesítése (17. táblázat) rávilágít, hogy a két vizsgált csoport hallgatóinak valamivel több, mint a harmada (75 fő) látta meg a kapcsolatot az egyik hibapár tagjai között, valamint, hogy a kitöltők közül 37 fő (18,23%) ismerte fel a kapcsolatot mindkét hibapár esetén. Fontosnak tartjuk kiemelni a 17. táblázat utolsó oszlopát, melyben a mindkét hibapár tagjait helyes kategóriába soroló tanulók százalékos aránya szerepel. Annak ellenére, hogy valamennyi hallgató a tesztet megelőzően pár hónappal sikeres érettségi vizsgát tett, a teljes minta 1,97%-a (négy fő) tudta a feladatot sikeresen elvégezni.

17. táblázat: A hibapárok megjelöléseinek százalékos összesítése a Gazdaságinformatikus (GI) és Programtervező informatikus (PTI) csoportok kitöltései alapján.

	Azonos hibatípus		**Helyes**	
	Egyik pár	**Mindkét pár**	**Egyik pár**	**Mindkét pár**
GI (%)	34,88	18,60	13,95	1,16
PTI (%)	38,46	17,95	30,77	2,56
Összesen (%)	36,95	18,23	23,65	1,97

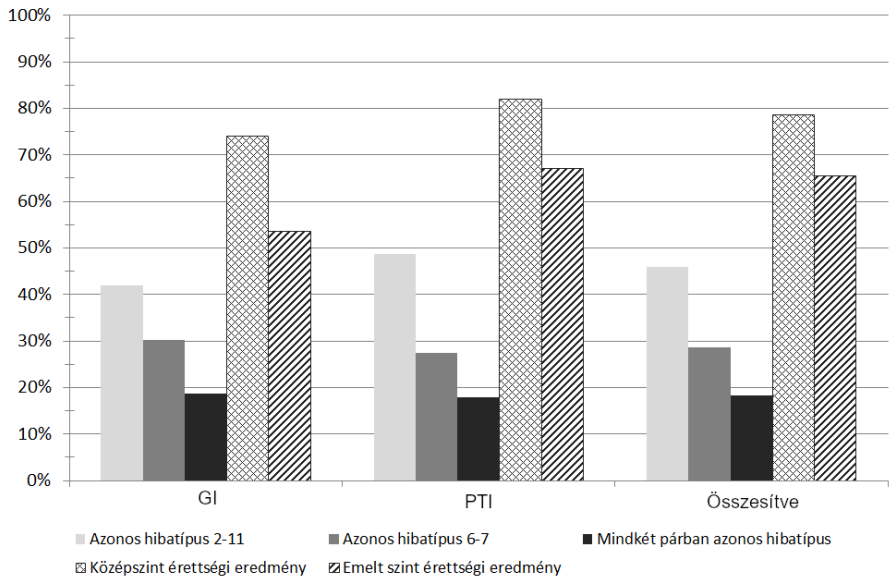
A vizsgálat második egységében a hallgatók által teszten elért eredményeinek és a középszintű érettségi eredményeik kapcsolatát elemeztük. A 18. táblázat adatai alapján látható az informatika érettségin elért eredményekhez viszonyítva a hallgatók valós tudása a szövegkezelési hibák felismerését illetően. A GI csoport hallgatóinak 18,60%-a látta meg az azonosságot mindkét hibapárban, amely 55,40%-os különbséget jelent az átlagos középszintű informatika érettségik eredményéhez képest.

A PTI csoport kitöltői közül 17,95% azonosította az összefüggéseket mindkét párban, amely 64,05% különbséget mutat a középszintű érettségi eredményekkel összevetve.

18. táblázat: A hibapárok megjelöléseinek százalékos összesítése, illetve az átlagos informatika érettségi eredmények a Gazdaságinformatikus (GI) és Programtervező informatikus (PTI) csoportokban.

	Azonos hibatípus				**Érettségi eredmény**	
	2–11	**6–7**	**Egyik pár**	**Mindkét pár**	**Közép szint**	**Emelt szint**
GI (%)	41,86	30,23	34,88	18,60	74,00	53,67
PTI (%)	48,72	27,35	38,46	17,95	82,00	67,13
Összesen (%)	45,81	28,57	36,95	18,23	78,73	65,60

A teljes minta 18,23%-a jelölte mindkét hibapár tagjait azonos kategóriába, amely 60,50%-ban tér el a csoportok átlagos középszintű informatika érettségi százalékától. Ismét hangsúlyoznánk, hogy az azonosságok felismerésének eredményei a hibapárok hibás, de azonos kategóriába sorolását is magukba foglalják. A hallgatók teljesítményét és az informatika érettségi eredményeiket a 22. ábra szemlélteti.



22. ábra: A csoportok (GI: Gazdaságinformatikus és PTI: Programtervező informatikus) százalékos eredményei az azonosságok felismerésében, valamint a közép- és emelt szintű informatika érettségin.

A vizsgált egységekbe foglalt eredmények alapján látható, hogy a hallgatók töredéke ismerte fel az azonosságokat akár egy hibapár esetében is. A csoportok teljesítménye a vizsgálat során változó tendenciát mutatott. A PTI csoport nagyobb arányban volt képes azonosítani az összefüggéseket az egyik hibapárban (+3,58%), azonban a GI csoport a mindkét párban felfedezett azonosságok arányában teljesített jobban (+0,65%). Az informatika érettségi százalékok tükrében a PTI csoport tagjai közép- (+8,00%) és emelt (+13,46%) szinten is jobb eredményekkel rendelkeztek, mint a GI csoport hallgatói. Ennek ellenére a két csoport teszten nyújtott teljesítménye között nem tapasztalhatunk jelentős eltérést, mely alapján kijelenthetjük, hogy a tanulók érettségi eredményei nem tükrözik a teszten elért százalékokat. Az eredmények, melyekből látható, hogy a hallgatók kevesebb, mint 50%-a ismert fel azonosságot bármelyik hibapár tagjai között azt a következtetést vonják magukkal, hogy a tanulók nem képesek meglátni a szöveges dokumentumok azonos szerkezeti egységei közötti összefüggéseket. Ez a hiányosság ellehetetleníti a jól formázott dokumentumok létrehozását. A [T3] tézis, amely kimondja, hogy az első éves informatika szakos hallgatók nem tudnak támaszkodni a közoktatásban elsajátított szövegkezelői ismereteikre, igazolt (Biró et al., 2015; Csapó, 2017c; Csernoch, 2009, 2010; Csernoch et al., 2015).

8. [T4] Programozási eszközök hatékonysága

8.1. Alkalmazott módszerek, eszközök és eljárások

Célunk a tézis vizsgálatával a szoftverfejlesztésre készült Construct 3 környezet által alkalmazott esemény-utasítás alapú vizuális nyelv hatékonyságának tesztelése volt, szemben az oktatási céllal létrehozott blokk-alapú vizuális programozási gyakorlattal a tanulók algoritmikus készségének és a számítógépes gondolkodásának fejlesztésére. A kutatási folyamatban a mintát egy helyi középiskola két 8. évfolyamos csoportja alkotta (vizsgálati és kontroll csoportok). A tanulók hat évfolyamos képzésben vettek részt és középiskolai tanulmányaik alatt nem tanulták korábban a programozás témakört. A két csoport azonos előrehaladást követve sajátította el a tudáselemeket, a vizsgálati csoport a Construct 3 környezetet, a kontroll csoport pedig a Scratch környezetet használva. A csoportok oktatása során kiemelt figyelmet fordítottunk arra, hogy a két környezet közötti különbségek minimálisan jelenjenek meg a kidolgozott feladatokban. Ennek megfelelően mindkét csoport azonos ütemtervet követve, azonos feladatokon dolgozott a saját fejlesztőkörnyezetükre optimalizálva. A vizsgálati csoport egy óra/hét, 18 héten keresztül, a kontroll csoport egy óra/hét, 17 héten keresztül tanulta a témakört.

Az adatgyűjtést nyomtatott tesztek formájában végeztük, melyeket a témakör előtt és után írtak meg a tanulók. Az előtesztek közvetlenül az első feladatmegoldás előtt kerültek megírásra, az utótesztek pedig a témakör lezárását követő első tanórán, késleltetés nélkül lettek kitöltve.

19. táblázat: A tanulók száma és a tesztek közötti eloszlás a vizsgálati és kontroll csoportokban.

	Vizsgálati csoport	**Kontroll csoport**
Tanulók száma	14	14
Előteszt	10	13
Utóteszt	12	13
Párosított tesztek	7	12

A minta kis méretét indokolja, hogy az iskola helyi tantervében a témakör csak kevés csoport esetében vált aktuálissá a tanévben. Továbbá, a programozás oktatására használt eszközök széles skálája miatt több csoport is inkompatibilis volt a jelen mérésben megszabott kritériumokkal. A korábban ismertetett, közismereti informatika tantárgy oktatásának jellemzői (5.1 fejezet) jelen esetben is fennálltak. A vizsgálati és a kontroll csoportok alacsony létszáma lehetővé tette számunkra a két csoport együtt haladásának biztosítását (19. táblázat). A teszt kitöltések közötti létszámingadozást (különös tekintettel a vizsgálati csoport párosított tesztjeinek esetében) a tanuló hiányzásokból, iskolai kirándulásokból és egyéb intézményi programokból eredő mulasztások indokolják.

Kiemelnénk, hogy a jelen mintában szereplő csoportok a [T1] és [T2] tézisekben tárgyalt csoportokkal együtt azonos intézményben tanultak. Ennek magyarázata, hogy kutatócsoportunknak az érintett iskolában volt lehetősége az oktatási folyamatokba bekapcsolódni és a vizsgálatokat elvégezni.

8.1.1. Tesztlapok bemutatása

A tanulók algoritmizálási készségének mérésére egy olyan tesztlap összeállítását tűztük ki célul, amely környezettől függetlenül, a már megalapozott Bebras (magyar megnevezésben e-Hód) feladatokra támaszkodik (ELTE IK T@T Labor, 2019; Pohl & Dagienė, 2019). A feladatokat úgy válogattuk össze, hogy azok a tanulók algoritmikus készségének szintjét több oldalról is vizsgálják, eltérő gondolkodást és megoldási stratégiát követelve. Megjegyeznénk, hogy a tesztlapban alkalmazott magyar nyelvű feladatok minimális eltéréseket mutatnak azok angol változataihoz képest, azonban ezek a különbözőségek nem befolyásolják a feladatok által mért készségeket.

A feladatok ismertetése során a sorszámuk helyett azok tesztlapon szereplő kódjával utalunk (Függelék 15.11). Az első feladat (*2012-HU-01A*) egy 3x3-as mátrixban történő értékek áthelyezésének nyomon követése volt négy lépésen keresztül. Ezt követte a *2015-CA-01* feladat, amely egy egyszerűsített gráfhasználati esetet vizsgált. A *2016-CH-12* feladatban a tanulóknak egy dekódolási problémát kellett

megoldaniuk grafikus elemeket (virágokat) értelmezve. A *2016-IS-02* feladat egy összetett feltételrendszert tárt a diákok elé, melyben egymásra épülő feltételeket elemezve kellett egy sorrendet felállítaniuk. A következő feladatban (*2016-CZ-03*) egy áttekinthető formában prezentált osztályhierarchia és -öröklődés esetet kellett megvizsgálni. Ezt követte egy szekvencialitást vizsgáló probléma (*2016-JP-03*), melyben a diákoknak egyszerű mozgató utasításokat (észak, dél, kelet és nyugat) kellett a megfelelő sorrendben végrehajtaniuk. A *2016-HU-06* feladat az objektumok csoportosítására és részhalmazok képzésére vonatkozó készségeket vizsgálta. A párhuzamosság elvét bemutató *2016-IE-05* feladatban a tanulók mozgató utasításokat használva irányítottak párhuzamosan három végrehajtó egységet, amelyekkel a kívánt output állapotot kellett elérniük. A következő feladat (*2016-LT-03*) egy ütemezési probléma megoldását várta el az utasítások megfelelő sorrendbe rendezésével. A *2016-NL-04* feladatban a tanulóknak egy számítógép által olvasható kódot (Kix kódot) kellett dekódolniuk négy karakter hosszúságban. Az utolsó feladat (*2016-SK-04*) egy egyszerűsített színkereső játék volt tippeléssel. A tanulók a tippekre kapott outputok alapján az input módosításával érték el a feladat által kért állapotot.

8.1.2. Adatok feldolgozása, elemzése

A tesztekben megadott adatokat saját kialakítású adatbázisban tároltuk. A feladatok jellegéből adódóan a tanulói válaszok szűk skálán mozogtak, több alkalommal is előfordult, hogy a megoldás egy betűjel vagy szám megadását jelentette. Ebből kifolyólag a feladatokra kapott válaszok elemekre bontása korlátozott volt. Kiemelnénk, hogy számos feladat előre megadott válaszopciókat kínált a tanulók számára. A tippelések kiküszöbölése végett azokban az esetekben, ahol a feladat jellege ezt lehetővé tette, az opciókat eltávolítva, nyílt kérdéseket fogalmaztunk meg.

Az adatok elemzésekor az egyes feladatok sajátosságaihoz igazodva indokolt esetben többletinformációkat tároltunk a tanulói válaszokról. Erre szolgál példaként a *2015-CA-01* feladat, melyre gyakori válasz volt az első ránézése a legtöbb kapcsolattal rendelkező karakter. Ebben az esetben a választ nem egyszerűen hibásnak mi-

nősítettük, hanem külön kategóriába soroltuk, hogy lehetőségünk legyen a két csoport közötti eltérés megfigyelésére a gyors gondolkodást megtévesztő választ illetően. Hasonló módon a *2016-JP-03* feladatra adott válaszok közül is elkülönítettük azokat a megoldásokat, amelyek célba juttatták a labdát, de tartalmaztak felesleges utasításokat is, illetve azokat, amelyek kitérő nélkül hibátlan megoldással oldották meg a problémát.

Az adatok elemzésére az R szoftvert használtuk (The R Foundation, 2019). A csoportok közötti különbségek elemzése, figyelembe véve a minta méretét, Fisher-féle egzakt teszttel történt.

8.2. Tézis vizsgálata

A két vizuális programozási megközelítés hatékonyságának vizsgálatához a tanulók utóteszten elért eredményeit elemeztük (20. táblázat). Az utótesztek vizsgálatát megelőzően összevettük a csoportok előtesztjeit, melyek alapján kijelenthetjük, hogy a vizsgálati és kontroll csoportok azonos tudásszinttel kezdték meg a témakör tanulását. A feladatokhoz kapcsolódó, utóteszteken elért részeredményeket a feladatok sorszáma helyett azok kódjával (8.1.1 fejezet) tüntetjük fel.

20. táblázat: Az utóteszt feladataira adott helyes és hibás válaszok száma a vizsgálati és kontroll csoportokban, illetve a csoportok közötti eltérés p-értékei.

	Helyes válaszok száma		**Hibás válaszok száma**		
Feladat	**Vizsgálati**	**Kontroll**	**Vizsgálati**	**Kontroll**	**p**
2012-HU-01A	10	11	2	2	1,0000
2015-CA-01	5	1	7	12	0,0730
2016-CH-12	12	13	0	0	
2016-IS-02	10	12	2	1	0,5930
2016-CZ-03	11	13	1	0	0,4800
2016-JP-03	12	11	0	2	0,4800
2016-HU-06	8	9	4	4	1,0000
2016-IE-05	11	10	1	3	0,5930
2016-LT-03	12	9	0	4	
2016-NL-04	11	10	1	3	0,5930
2016-SK-04	11	13	1	0	0,4800

A *2012-HU-01A* feladatot közel minden tanuló helyesen oldotta meg. Ezzel szemben a *2015-CA-01* feladat már mindkét csoport számára nehézségeket okozott. A tanulók nagy számban nevezték meg a legtöbb közvetlen kapcsolattal rendelkező hibás válaszlehetőséget („Gila”), amely arra utal, hogy a megtévesztő első benyomás alapján válaszolták meg a feladatot annak továbbgondolása nélkül. A helyes válaszok tekintetében a vizsgálati csoport jobban teljesített ebben a feladatban öt jó válasszal szemben a kontroll csoport egy helyes megoldásával. Érdekesség, hogy a helyes válaszok aránya nem változott az előtesztben tapasztaltakhoz képest, azonban mindkét csoport nagyobb arányban jelölte meg az említett („Gila”) hibás válaszlehetőséget: vizsgálati csoport három fővel többen, kontroll csoport négy fővel többen. A *2015-CA-01* feladatot minden tanuló hibátlanul megoldotta, ezért annak eredményeit és a csoportok közötti eltérést a Fisher-féle egzakt teszttel nem tudtuk vizsgálni. A *2016-IS-02* és *2016-CZ-03* feladatokat a csoportok közel minden tanulója helyesen válaszolta meg, minimális eltéréssel a hibás válaszok számát illetően. A *2016-JP-03* feladatra is nagy arányban érkeztek helyes válaszok. Fontos kiemelni, hogy az előteszt adatainak elemzésekor kialakult gyakorlat szerint a feladatra érkezett megoldások közül azokat is helyesnek fogadtuk el, amelyek végeredményként célba juttatták a labdát, függetlenül attól, hogy a tanuló mennyi felesleges utasítás használatával valósította ezt meg. A helyes válasz kategóriába sorolt megoldások közül az utóteszten a vizsgálati és kontroll csoportokból is 11-11 fő oldotta meg a feladatot felesleges utasítás használata nélkül. A *2016-HU-06* feladat megoldása során a hibás választ adó tanulók rendszeresen a D és E betűjellel ellátott edényeket jelölték meg, amely ismételten a feladat felszínes értelmezésével, annak gyors gondolkodással történő megoldásával magyarázható. A *2016-IE-05* feladatot szintén közel minden tanuló hibátlanul oldotta meg. A *2016-LT-03* feladat hasonló aránnyal rendelkezik a helyes válaszok számát illetően, azonban a feladat feldolgozása során a megfelelő sorrendbe helyezett válaszok egyes eseteit külön kezeltük. Ebből kifolyólag a kapott eredmények elemzése a Fisher-féle egzakt teszttel nem volt értelmezhető. A vizsgálati csoport minden tagja hibátlanul oldotta meg a feladatot, míg a kontroll csoport-

ban kilenc fő nyújtott hibátlan választ és további négy fő részlegesen helyes megoldással teljesítette azt. A Kix kódra épülő dekódolás *2016-NL-04* feladatban a helyes válaszok kategóriájába soroltuk azokat az eseteket is, amelyekben a tanuló a kódnak csak egy részletét tudta megfejteni. A hibátlan megfejtések számát tekintve a vizsgálati csoport tíz, a kontroll csoport pedig nyolc megoldást számlál. A teszt utolsó, *2016-SK-04* feladatára egy fő kivételével minden tanuló hibátlan választ adott.

A fenti eredményekből látható, hogy a tesztbe foglalt feladatokra nagy arányban érkeztek helyes válaszok mindkét csoport esetében. A csoportok közötti eltérések szignifikancia vizsgálatának tekintetében a 20. táblázat alapján kijelenthető (a két kiemelt feladattól eltekintve), hogy a vizsgálati és kontroll csoportok között egy esetben sem merül fel szignifikáns különbség. Ez alól az eredményekben legnagyobb eltérést mutató *2015-CA-01* feladat sem kivétel ($p = 0{,}0730$).

Összefoglalva, a vizsgálati és kontroll csoport eredményeit illetően nem mutatkozott szignifikáns különbség az algoritmikus készséget és a számítógépes gondolkodást mérő Bebras feladatok (ELTE IK T@T Labor, 2019; Pohl & Dagienė, 2019) megoldásaiban. Ebből következve a [T4] tézis, mely alapján a Construct 3 eseményutasítás alapú vizuális programozás hatékonyabban fejleszti a tanulók algoritmikus készségét a Scratch blokk-alapú vizuális nyelvénél, elvetett (Csapó, 2019; Csernoch et al., 2015; Fincher et al., 2010; Franklin et al., 2016; Kalelioglu & Gülbahar, 2014; Papadakis et al., 2014). Az eredmények publikálása a közeljövőben várható.

9. Összefoglaló

Napjaink informatikával átszőtt társadalmában a legfontosabb készségek közé soroljuk az algoritmikus készséget és a számítógépes gondolkodást, melyek fejlesztése elsődlegesen az informatikaoktatásra hárul. Ezekhez a készségekhez és a mindennapi élethez szorosan kapcsolódik az információk rögzítése, kezelése és feldolgozása, leggyakrabban irodai szoftvercsomagok segítségével. A gyakorlatban azonban a felhasználók sok esetben abban a téves hitben használják ezeket a szoftvereket, hogy munkájuk során tudatosan, kellő hozzáértéssel és megfelelő algoritmusok szerint felépítve állítják elő a dokumentumokat. Ennek eredményeképp számos hibás produktum lát napvilágot, melyek potenciális veszélyeket hordoznak magukban (3.3 fejezet). Az informatikaoktatásnak szerves részét képezi az alkalmazói ismeretek oktatása, azonban annak puszta jelenléte a tantervekben – látva a hibás dokumentumok nagyszámú előfordulását – nem garantálja a jól formázott és hibamentes dokumentumok létrehozásának gyakorlatát. Ebből kiindulva indokoltnak láttuk alternatív megközelítések vizsgálatát a táblázatkezelés és szövegszerkesztés témakörök oktatására.

Az algoritmikusok felépítésével az informatikaoktatásban elsődlegesen a programozás témakörében találkoznak a tanulók, amely a jelenlegi gyakorlatban meglehetősen változatos formában jelenik meg. Korábbi kutatások eredményei alapján a közoktatásból kikerülő tanulók nem rendelkeznek olyan szintű algoritmikus készséggel és számítógépes gondolkodással, amelyre hatékonyan támaszkodhatnának. A tanulók ezen készségeinek fejlesztésére a programozás témakörben egy alternatív vizuális programozásra támaszkodó, elsődlegesen szoftverfejlesztési céllal megalkotott környezet bevezetését és vizsgálatát láttuk indokoltnak.

Munkánk során a táblázatkezelés, szövegszerkesztés és programozás témakörök hatékonyabb oktatására szolgáló alternatív módszerek vizsgálatával foglalkoztunk, melyek alkalmasak nem tradicionális programozási környezeteket és megközelítéseket használva az algoritmikus készség és számítógépes gondolkodás fejlesztésére.

A Spreadsheet Lego (Sprego) módszertan algoritmikus szemléletet előtérbe helyezve, sémák kialakításán keresztül oktatja a táblázatkezelés témakört (3.5.1 fejezet). A feladatmegoldás során a tanulók Pólya négylépéses problémamegoldó stratégiájára alapozva (Pólya, 1954) autentikus forrásokat használva alapvető algoritmusokkal dolgoznak általános célú táblázatkezelői függvények segítségével kódolva azokat.

A szövegkezelés oktatására az Error Recognition Model (ERM) módszertannal dolgoztunk. Az ERM – hasonlóan a táblázatkezelés témakörben alkalmazott Sprego módszertanhoz – autentikus forrásokra támaszkodik és az interneten nagy mennyiségben fellelhető hibás szöveges dokumentumok elemzésén és kijavításán keresztül oktatja a szövegkezelési ismereteket (3.5.2.1 fejezet).

A vizuális programozási nyelvek elterjedt változatai közül az esemény-utasítás alapú nyelvek táborát bővíti a Construct 3 környezet megközelítése (3.5.3.5 fejezet). A tanulók a Construct 3 segítségével grafikus megjelenésű alkalmazásokat készíthetnek, melyek vizuális kódját eseményblokkok definiálásával építik fel. A környezet elsődlegesen ipari szoftverfejlesztési célokra készült azzal a megközelítéssel, hogy lehetőséget biztosítson minimális korlátozásokkal a programozásra, fejlesztői tapasztalattal nem rendelkező emberek számára is.

9.1. Tézisek eredményei

9.1.1. [T1] Spreadsheet Lego hatékonyságvizsgálata

A tézisbe foglalt vizsgálat során teszteltük a Sprego módszertan hatékonyságát a táblázatkezelés témakör oktatására szemben a tradicionális felületi megközelítésekkel (5.1 fejezet). A mintába egy helyi középiskola két 7. évfolyamos (vizsgálati és kontroll) és egy 10. évfolyamos (vizsgálati) csoportjai tartoztak. A vizsgálati csoportok a Sprego módszertannal tanulták a témakört, a kontroll csoport pedig tradicionális felületi megközelítés segítségével. A tanulók a témakört megelőzően előtesztet, illetve azt követően utótesztet töltöttek ki. A vizsgálatot három fő egységre bontottuk (5.2 fejezet).

Az első egységben a két 7. évfolyamos csoport (vizsgálati és kontroll) utótesztjeinek eredményeit elemeztük. A két csoport közötti különbség alapján a Sprego szignifikánsan hatékonyabbnak bizonyult a táblázatkezelés témakör oktatására, mint a tradicionális megközelítések (6. táblázat). A vizsgálati csoport 65,42%-os eredményt ért el, szemben a kontroll csoport 38,00%-os eredményével ($p = 0,0013$). Ebből következik, hogy a Sprego módszertannal tanult csoport tagjai szignifikánsan hatékonyabban tudtak algoritmusokat építeni. Ezzel szemben a kontroll csoportnak nehézségei adódtak a probléma-specifikus függvények használatával, illetve a mögöttük meghúzódó algoritmusok megértésével.

A vizsgálat második egységében a 7. és 10. évfolyamos vizsgálati csoportok utótesztjeinek eredményeit hasonlítottuk össze. Az adatok elemzése alapján azok a tanulók, akik korábban probléma-specifikus függvények használatával tanulták a témakört, nem tudtak jobb eredményt elérni azoknál a diákoknál, akik először találkoztak táblázatkezeléssel a Sprego módszert használva (20. ábra). A 7. évfolyamos csoport 65,42%-ot, a 10. évfolyamos csoport pedig 64,34%-ot ért el a teszten ($p = 0,7388$).

A harmadik egységben a 10. évfolyamos tanulók problémamegoldási gyakorlatát elemeztük, hogy milyen arányban alkalmazzák a Sprego módszertan képletépítési megközelítését, illetve a korábban tanult problémaspecifikus függvényeket. A tanulók megoldásai alapján a csoport tagjai váltogatnak a két megoldási megközelítés között. Más megfogalmazásban: azok a tanulók, akik korábban probléma-specifikus függvényeket, majd később a Sprego módszertant használták, igyekeznek megoldásaikat a Sprego módszertannal kialakított sémák alapján felépíteni miközben a folyamatot erősen befolyásolja az előzetes tudás.

Az eredményeket figyelembe véve kijelenthetjük, hogy a Sprego módszertan számos előnnyel rendelkezik a hagyományos megközelítésekkel szemben, valamint, hogy szignifikánsan hatékonyabb a témakör oktatására.

9.1.2. [T2] Spreadsheet Legoval hosszú távú tudás kialakítása

A második tézisben foglaltak értelmében a Sprego módszertannal történő hosszú távú tudás kialakításának hatékonyságát vizsgáltuk (6.1 fejezet). A mintába egy helyi középiskola hét csoportja tartozott, melyből két 8. évfolyamos és két 10. évfolyamos csoport alkotta a vizsgálati csoportokat és három 10. évfolyamos csoport a kontroll csoportokat. A tanulók mindegyike az előző tanévben tanulta utolsó alkalommal a táblázatkezelés témakört (a vizsgálati csoportok a Sprego módszertannal, a kontroll csoportok tradicionális megközelítésekkel). A diákok a vizsgálat keretein belül késleltetett utóteszt formájában adtak számot tudásukról. Az eredmények alapján azok a tanulók, akik a vizsgálatot megelőző évben Sprego módszertannal tanulták a témakört, jobban teljesítettek a teszteken, mint a kontroll csoport tanulói (6.2 fejezet; 13. táblázat). A vizsgálati csoportok összesített 48,20%-os eredménye összevetve a kontroll csoportok összesített 19,37%-os eredményével szignifikáns eltérést mutatott ki ($p = 0{,}0000$).

A tézis további alátámasztására szolgál a [T1] tézis vizsgálatában szereplő előtesztek eredményeinek elemzése (14. táblázat). A 7. évfolyamos tanulók (vizsgálati és kontroll csoportok egybefogva) előzetes tudás hiányában a teszteket 5,52%-os átlagos eredménnyel teljesítették, melyhez képest a 10. évfolyamos diákok 18,75%-os összesített eredménye szignifikáns eltérést mutat ($p = 0{,}0000$). Ezt az eltérést a 10. évfolyamos tanulók azonban 8% alatti teljesítménnyel érték el a teszt képletalkotási és függvényhasználati részében. Figyelembe véve a 10. évfolyam előzetes tudását, miszerint korábban két évfolyamon keresztül már tanultak táblázatkezelést, a hagyományos megközelítésekkel kialakított hosszú távú tudás hatékonysága nem alátámasztott. A 10. évfolyamos csoport tagjai nem tudtak támaszkodni korábbi tudásukra a teszt kitöltése során.

A tézisbe foglalt vizsgálat eredményei alapján a Sprego módszertan szignifikánsan hatékonyabb a hosszú távú tudás kialakításában, mint a tradicionális módszerek.

9.1.3. [T3] Dokumentumelemzés és hibafelismerés

A harmadik tézisben a középiskolát elvégzett tanulók dokumentumelemző és hibafelismerő készségeit vizsgáltuk a szövegkezelés témakörben (7.1 fejezet). A mintát a Debreceni Egyetem Informatikai Karára 2014-ben Gazdaságinformatikus (GI) és Programtervező informatikus (PTI) alapképzésekre felvett hallgatók alkották. A hallgatók felsőfokú tanulmányaik első hetében kitöltötték a TAaAS felmérést, melyből a szövegkezelés feladatok hibafelismerő részét elemeztük. A vizsgálatot két egységre bontottuk (7.2 fejezet).

Az első egységben azt elemeztük, hogy a megadott mintadokumentumban az azonos hibakategóriába tartozó hibapárok tagjai között a hallgatók meglátták-e az összefüggéseket. Az eredmények alapján a két csoport tagjai közel azonos arányban tudták a megjelölt hibapárok közül az egyikben meglátni az összefüggéseket (GI: 34,88%, PTI: 38,46%, összesítve: 36,95%) (17. táblázat). A csoportok eredményei a következőképp alakultak a mindkét vizsgált hibapárban meglátott kapcsolatokat illetően: GI: 18,60%, PTI: 17,95%, összesítve: 18,23%.

A vizsgálat második egységében a tanulók teszten elért eredményeit hasonlítottuk össze az informatika érettségi eredményeikkel (18. táblázat). A teljes minta 36,95%-a fedezte fel az egyik hibapár tagjai közötti kapcsolatokat, illetve 18,23%-a látta meg az összefüggést mindkét hibapár esetében. Ezzel szemben a tanulók összesített középszintű informatika érettségi eredménye 78,73%, valamint emelt szinten 65,60% volt. Az adatokból látható, hogy az informatika érettségi két szintjén elért eredmények nem tükrözik a tanulók teszten elért százalékait (22. ábra).

A vizsgálat eredményei alapján a közoktatást elvégző tanulók nem rendelkeznek a dokumentum szerkezeti egységei közötti kapcsolatok felismeréséhez szükséges készségekkel, melyek hiányában nem lehetséges jól formázott dokumentumok előállítása.

9.1.4. [T4] Programozási eszközök hatékonysága

A negyedik tézis vizsgálata során arra kerestük a választ, hogy programozás témakörben különféle vizuális programozási környezeteket és nyelveket használó tanulói csoportok algoritmikus készsége és számítógépes gondolkodása között milyen eltérések tapasztalhatók. A mintába egy helyi középiskola két 8. évfolyamos csoportja tartozott, melyek párhuzamos előrehaladással tanulták a programozás témakört. A vizsgálati csoport a Construct 3, a kontroll csoport pedig a Scratch szoftverrel dolgozott (8.1 fejezet). Az adatok gyűjtését algoritmikus készséget vizsgáló feladatokra alapozott tesztek segítségével végeztük (8.1.1 fejezet). A vizsgálat eredményei alapján (20. táblázat) a két csoport közel azonos szinten teljesítette az utóteszteket, a feladatokat nagy arányban helyesen megoldva (8.2 fejezet). A vizsgálati és kontroll csoportok közötti eltérések vizsgálata alapján nem mutatható ki szignifikáns különbség a tanulók algoritmikus készsége között.

A tézis vizsgálata alapján a szoftverfejlesztési céllal létrehozott Construct 3 környezet alkalmazása nem fejleszti hatékonyabban a tanulók algoritmikus készségét, mint az oktatási céllal készült Scratch környezet.

9.2. Jövőbeli elképzelések

A későbbiekben a jelen dolgozatban ismertetett Spreadsheet Lego módszertanra alapozott vizsgálatok eredményeit további kutatások keretein belül is megvizsgáljuk. Bár a módszertan hatékonyságvizsgálata pozitív eredményekkel zárult, a minta méretéből adódóan további vizsgálatok indokoltak. Ebből kiindulva tervezzük a módszertan hatékonyságvizsgálatát nagyobb tanulói mintán, kiegészítve az általunk fejlesztett oktatószoftverek hatékonyságának elemzésével a táblázatkezelés témakör oktatására.

További terveink között szerepel az Error Recognition Model módszertan hatékonyságvizsgálata középfokú informatikaoktatásban a szövegszerkesztés témakör tanítására, a hagyományos felületi megközelítésekkel szemben.

Szeretnénk a programozásoktatás során alkalmazott eszközök algoritmikus készség és számítógépes gondolkodás fejlesztési hatékonyságának további vizsgálatát is elvégezni nagyobb tanulói mintán, változatosabb eszközkészletet elemezve. További szempontként szerepel az adatgyűjtés során alkalmazott tesztlapok újratervezése a tanulói válaszok és a mért készségek szélesebb spektrumának bevezetése céljából.

10. Summary

In today's computer driven society, algorithmic and computational thinking skills are considered among the most important skills. The development of these skills primarily relies on ICT (Information and Communications Technology) education. Information handling and processing mainly carried out using birotical (office) software are connected to these skills and are required in our everyday life, as well. However, in practice most end users believe that they use these software programmes consciously, with adequate expertise, and that they are building their documents using correct and well-structured algorithms. This results in a large number of bricolage documents, which carry potential risks (section 3.3). Teaching end-user applications receives great emphasis in ICT education. However, seeing the frequent occurrences of bricolage documents, the mere presence of these topics in the curricula does not ensure the practice of creating well-formatted and error-free documents. Based on this situation, an analysis of alternative methods for teaching the spreadsheet- and text-management topics is required.

The creation of algorithms in ICT education is mostly present in programming, which presents itself in a diverse form regarding the tools used. The use of text-based programming languages, the various environments developed with educational usage in mind, and also other approaches can be found at different levels of ICT education. Prior research show that students who complete secondary education do not possess the required level of algorithmic and computational thinking skills on which they can build effectively. Developing these skills in the area of programming motivated us to introduce and analyse an alternative visual programming environment developed for software production.

In our work we researched alternative approaches and methodologies for teaching spreadsheet-management, text-management, and programming more effectively, using non-traditional programming environments for developing algorithmic and computational thinking skills.

The Spreadsheet Lego (Sprego) methodology teaches spreadsheet-management from an algorithmic point of view (section 3.5.1). The problem-solving process of Sprego follows on the four-step problem-solving strategy of Pólya (1954). The method builds on a small number of general-purpose functions instead of the large number of problem-specific functions built into spreadsheet-management environments. The students build and code the algorithms for solving problems step-by-step using composite array formulas. In this process the methodology puts great emphasis on planning these steps, with their input and output values highlighted. Working and learning through this methodology is supported by unplugged (a 3D printed doll set, origami boats) and semi-unplugged tools. As a semi-unplugged tool, we developed an educational application (section 3.5.1.2) to visually present the algorithms of the most common Sprego problems. As another example of semi-unplugged tools, the method uses authentic sources. These sources are webtables converted from the internet for classroom use and are based on real-life data. Students work with these tables (leaving the unnecessary practice of typing tables behind) and solve problems that are present in their everyday lives.

For teaching text-management we worked with the Error Recognition Model (ERM) methodology. The ERM – similarly to the Sprego methodology used in spreadsheet-management – works with authentic sources. It teaches the topic through document analysis and correction using the large number of erroneous documents that can be found on the internet (section 3.5.2.1). The students analyse the documents, recognize, categorize and correct the errors in them, and during this practice-oriented workflow they learn the required knowledge items for text-management. The goal of the ERM methodology is to make students learn how to create well-formatted documents and understand the algorithmic process of constructing the structure of text documents by the end of the topic. The method uses the printed form of the documents as unplugged tools, which separate the errors recognizable in printed and in digital form while also helping students follow the process of the error correction.

The Construct 3 environment uses an event-action based visual programming method (section 3.5.3.5). Students working with Construct 3 create graphical applications whose visual codes are built by defining event blocks. The event blocks first test the conditions encompassed in them, and when the conditions are met, they run the connected actions. The environment is built with software production in mind with an approach to make programming possible with minimal limitations for users who have no coding experience. Solving problems with Construct 3 focuses on planning and implementing algorithms. In the first step students decompose the problem into subtasks with teacher guidance. After selecting the goal, they build the algorithm required for solving the problem, highlighting their input and output values. This is followed by the definition of the visual code using natural language, which is then coded, tested and if necessary, corrected. Students receive immediate and spectacular feedback on their work which can be shared with other students using the functionalities of the environment. Although Construct 3 mainly focuses on the development of multimedia applications and games, it also offers the opportunity to solve traditional programming problems.

10.1. Results of the theses

10.1.1. [T1] The effectiveness of the Spreadsheet Lego methodology

We conducted a measurement to test the effectiveness of the Sprego methodology for teaching the spreadsheet-management topic compared to traditional surface approaches (section 5.1). Regarding the sample, we worked with two grade-7 (experimental and control) and one grade-10 groups (experimental) from a local high-school. The experimental groups studied the topic with Sprego, while the control group used traditional methods. The students completed a pre- and a post-test before and after the spreadsheet management topic. We divided the research into three main parts (section 5.2).

In the first part we compared the two grade-7 groups' results (experimental and control) on the post-tests. Based on the differences between the two groups, Sprego

proved to be significantly more effective in teaching spreadsheet-management than the traditional approaches, which do not prepare students for data-handling and conscious algorithm creation (table 6). The experimental group scored 65.42% compared to the 38.00% of the control group ($p = 0.0013$). It follows that the Sprego group can construct algorithms significantly more effectively. In contrast, the control group faced difficulties using the problem-specific functions and understanding the algorithms behind them.

In the second part of the analysis we compared the post-test results of the grade-7 and grade-10 experimental groups. Based on their prior knowledge, we expected the grade-10 group to have difficulties in their progression: that they will not achieve significantly higher scores in the post-tests than the grade-7 students. The examination of the data proved our expectation was correct. We found that students who previously studied the topic using problem-specific functions could not achieve better results than students who started learning spreadsheet management with Sprego for the first time (figure 20). The grade-7 group scored 65.42%, while the grade-10 students completed the test with a result of 64.34% ($p = 0.7388$).

Regarding the third part of our research, we analysed the problem-solving practice of the grade-10 students: how students who have studied the topic with both traditional approaches and with Sprego choose items from their function base and with what approach they construct their algorithms. Based on the students' solutions, the members of the grade-10 group alternate between the two strategies. In other words, the students who have used problem-specific functions prior to their Sprego classes try to construct their solutions based on the schemata they developed in Sprego classes while the process is strongly influenced by their prior knowledge.

Based on the results, we can conclude that the Sprego methodology has advantages over the traditional approaches, and that it is significantly more effective for teaching the topic.

10.1.2. [T2] Developing long-term knowledge with Spreadsheet Lego

In our second thesis we examined the effectiveness of the Sprego methodology for developing long-term knowledge (section 6.1). We included seven groups of a local high school in our sample: two grade-8 and two grade-10 groups formed the experimental groups, while the control groups consisted of three grade-10 groups. All the students had learned spreadsheet-management for the last time one year prior to our research (the experimental groups used Sprego, while the control groups studied with traditional approaches). We collected data using delayed post-tests. Based on the results, students who had learned spreadsheet management with Sprego completed the tests more successfully than the members of the control groups (section 6.2; table 13). The 48.20% result of the experimental groups compared to the 19.37% score of the control groups resulted in a significant difference ($p = 0.0000$).

The analysis of the pre-tests of the research detailed in the [T1] thesis further strengthens our current thesis (table 14). The grade-7 students (experimental and control groups combined) completed the tests with a score of 5.52%, reflecting their lack of prior knowledge. This compared with the grade-10 students' score of 18.75%, resulting in a significant difference ($p = 0.0000$). However, the grade-10 students achieved this difference with extremely low scores on the algorithm construction and formula creation part of the test (below 8%). Considering the prior knowledge of the grade-10 students, who had studied the topic for two years beforehand, the effectiveness of the long-term knowledge developed with traditional approaches is unsupported. The grade-10 students could not build on their previous knowledge when they completed the test.

Based on the results of the analysis in the current thesis, the Sprego methodology is significantly more effective in developing long-term knowledge compared to traditional methods.

10.1.3. [T3] Document analysis and error recognition

In our third thesis we examined the document analysis and error recognition abilities of students who had completed their secondary education (section 7.1). The sample consisted of first year undergraduates who had enrolled at the University of Debrecen, Faculty of Informatics in 2014 for Business Informatics BSc (BI) and Computer Science BSc (CS) majors. The undergraduates completed the TAaAS test during the first week of their tertiary education, and we analysed the error recognition part of the text-management tasks. We separated the analysis into two main parts (section 7.2).

In the first part we examined whether students recognised the connections between members of error-pairs in the same category in the sample document provided. In other words, we wanted to understand whether the students possessed the required algorithmic and computational thinking skills to recognise the connections between similar structural elements of a text document. Regarding the results, a similar proportion of the undergraduates in the two groups recognised the connection between members of one error-pair (BI: 34.88%, CS: 38.46%, summarized: 36.95%) (table 17). The results of the groups changed as we included the second error-pair in our analysis: BI: 18.60%, CS: 17.95%, summarized: 18.32%. If, in addition to recognizing the connections between the error-pairs, we included the sorting of members into the correct error category (as the task on the test-sheet required) we could observe another drop in the students' score: BI: 1.16%, CS: 2.56%, summarized: 1.97%.

In the second part of our research we compared the students' results on the tests with the overall score of their computer science graduation exams (table 18). 36.95% of the members of the whole sample recognised the connection between members of one error-pair, while for both error-pairs the proportion was 18.23%. In contrast, the students' summarized results of their computer science graduation exams were 78.73% at the intermediate level and 65.60% at the advanced level. These results show that the scores acquired at either level of the aforementioned exams do not

reflect the students' actual abilities in recognising the connections between similar error occurrences (figure 22).

Based on the results, students who complete secondary education do not possess the required skills to recognize the connections between different structural parts of a document. Without this ability the creation of well-formatted documents is impossible.

10.1.4. [T4] The effectiveness of the programming tools

While analysing the fourth thesis we looked for differences between the level of algorithmic and computational thinking skills of students who used differing visual programming environments throughout the programming topic. Regarding the sample, we analysed two grade-8 groups from a local high-school who both studied and progressed on the programming topic at a similar pace. The experimental group worked with Construct 3, while the control group with Scratch (section 8.1). The data collection was carried out using test-sheets consisting of tasks designed to measure the students' algorithmic skills (section 8.1.1). The results show (table 20) that the two groups completed the post-tests at the same level with minimal deviations, correctly answering most of the tasks (section 8.2). Analysing the differences between the experimental and control groups, no significant difference is found between the algorithmic skills of the students.

Based on the analysis of the thesis, using the Construct 3 environment created with software production in mind could not develop students' algorithmic skills more effectively than the Scratch environment designed for education.

10.2. Future perspectives of the research

In our future work we plan on continuing the analysis of the Spreadsheet Lego methodology presented in this dissertation. Although examining the effectiveness of the methodology provided positive results, further measurements are advised based on the size of the sample. Following on from this, we plan to continue testing its

effectiveness on a larger sample of students, along with the effectiveness of our educational software for teaching spreadsheet management.

We also aim to test the effectiveness of the Error Recognition Model in secondary ICT education for teaching text-management, compared to the currently used traditional surface approaches.

We would like to further analyse the effectiveness of the various programming tools used in programming classes for developing students' algorithmic and computational thinking skills. We plan on conducting this research on a larger sample of students along with the examination of more programming tools. We are also considering the re-design of the test used in data collection to introduce a wider array of student answers and tested skills.

11. Köszönetnyilvánítás

Szeretném köszönetemet és hálámat kifejezni témavezetőmnek, Dr. Csernoch Máriának az egyetemi tanulmányaim alatt nyújtott folyamatos és fáradhatatlan segítségéért, iránymutatásáért és támogatásáért.

Kutatásom és munkám során a támogatásért és a hasznos tanácsokért szeretném köszönetemet kifejezni programvezetőmnek, Dr. Terdik György professzor úrnak.

Továbbá köszönetet szeretnék mondani tanáraimnak és kollégáimnak, főként Abari Kálmánnak, aki odaadó segítségével és szakértelmével segítette kutatásaim létrejöttét. További köszönet illeti hallgatótársamat és barátomat, Sebestyén Katalint, akivel egymást támogatva és segítve megvalósíthattuk közös projektjeinket. Köszönetet szeretnék mondani továbbá a kutatócsoportunk többi tagjának, családomnak és barátaimnak, akik segítettek, bátorítottak és támogattak munkámban és előre haladásomban, illetve azoknak a tanároknak és tanulóknak, akik bekapcsolódtak a kutatásomba és lehetővé tették annak megvalósulását.

12. Irodalomjegyzék

ACM & IEEE (2013). Computer Science Curricula: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science. ACM Press, and IEEE Computer Society Press, New York. DOI=http://dx.doi.org/10.1145/2534860.

Aleksić, V. & Ivanović, M. (2016). Introductory Programming Subject in European Higher Education. Informatics in Education, 2016, 15(2), pp. 163–182.

Anderson, D. L. (2017). Improving information technology curriculum learning outcomes. Informing Science: the International Journal of an Emerging Transdiscipline, 2017, 20, pp. 119–131.

Angeli, C. (2013). Teaching Spreadsheets: A TPCK Perspective. Improving Computer Science Education, 2013, pp. 132–145. Routledge, New York and London.

Baranyi, P. & Gilanyi, A. (2013). Mathability: emulating and enhancing human mathematical capabilities. 4th IEEE International Conference on Cognitive Infocommunications, CogInfoCom, 2013, pp. 555–558. DOI=http://doi.org/10.1109/CogInfoCom.2013.6719309. IEEE.

Bell, T. & Newton, H. (2013). Unplugging Computer Science. D. M. Kadijevich, C. Angeli, & C. Schulte, Improving Computer Science Education, pp. 75–90. Routledge.

Ben-Ari, M. & Yeshno, T. (2006). Conceptual models of software artifacts. Interacting with Computers, 2006, 18(6), pp. 1336–1350. DOI=http://doi.org/10.1016/j.intcom.2006.03.005.

Ben-Ari, M. (1999). Bricolage Forever!. PPIG 1999, 11th Annual Workshop, University of Leeds: Computer-Based Learning Unit, UK, 5–7 January, 1999.. http://www.ppig.org/papers/11th-benari.pdf. Letöltés dátuma: 2016.12.01.

Ben-Ari, M. (2011). Non-myths about programming, Communications of the ACM, 2011, 54(7), pp. 35. DOI=http://doi.org/10.1145/1965724.1965738.

Berki, B. (2018). Desktop VR and the Use of Supplementary Visual Information. 9th IEEE International Conference on Cognitive Infocommunications, CogInfoCom, 2018, pp. 333–336. DOI=http://doi.org/10.1109/CogInfoCom.2018.8639925. IEEE.

Biró, P. & Csernoch, M. (2013). Elsőéves informatikushallgatók algoritmizáló készségei. XXIII. Nemzetközi Számítástechnika és Oktatás Konferencia, SzámOkt 2013, EMT, pp. 154–159.

Biró, P. & Csernoch, M. (2015a). The mathability of computer problem solving approaches. 6th IEEE International Conference on Cognitive Infocommunications, CogInfoCom, 2015, pp. 111–114. DOI=http://doi.org/10.1109/CogInfoCom.2015.7390574. IEEE.

Biró, P. & Csernoch, M. (2015b). The mathability of spreadsheet tools. 6th IEEE International Conference on Cognitive Infocommunications, CogInfoCom, 2015, pp. 105–110. DOI=http://doi.org/10.1109/CogInfoCom.2015.7390573. IEEE.

Biró, P. & Csernoch, M. (2017a). Unplugged tools for building algorithms with Sprego. International Conference on Education and New Development, END2017, 2017, pp. 401–405.

Biró, P. & Csernoch, M. (2017b). Semi-unplugged tools for building algorithms with Sprego. The Turkish Online Journal of Educational Technology, INTE 2017 November, pp. 946–957. http://tojet.net/special/2017_11_2.pdf. Letöltés dátuma: 2017.12.11.

Biró, P., Csernoch, M., Máth, J. & Abari, K. (2015). Measuring the level of algorithmic skills at the end of secondary education in Hungary. Procedia - Social And Behavioral Sciences, 2015, 176, pp. 876–883.

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

Bujdosó, Gy. & Csernoch, M. (2014). Digitális írástudás, digitális nyelvhelyesség. Tudományos és Műszaki Tájékoztatás, 2014, 61(10), pp. 359–377.

Carr, N. (2011). The shallows: What the Internet is doing to our brains. WW Norton and Company, New York.

Carretero, S., Vuorikari, R. & Punie, Y. (2017). The Digital Competence Framework for Citizens. Publications Office of the European Union.

Case, J. M. & Gunstone, R. F. (2002). Metacognitive development as a shift in approach to learning: an in-depth study. Studies in Higher Education, 2002, 27(4), pp. 459–470.

Case, J. M. & Gunstone, R. F. (2003). Approaches to learning in a second year chemical engineering course. International Journal of Science Education, 2003, 25(7), pp. 801–819.

Champagne, A. B., Gunstone, R. F. & Klopfer, L. E. (1983). Naive knowledge and science learning. Research in Science and Technological Education, 1983, 1(2), pp. 173–183.

Chen, J. A., Morris, D. B. & Mansour, N. (2015). Science Teachers' Beliefs. Perceptions of Efficacy and the Nature of Scientific Knowledge and Knowing. H. Fives & M. G. Gill, International Handbook of Research on Teachers' Beliefs, Routledge, pp. 370–386.

Chmielewska, K., Gilányi, A. & Łukasiewicz, A. (2016). Mathability and Mathematical Cognition. 7th IEEE International Conference on Cognitive Infocommunications, CogInfoCom, 2016. DOI=http://doi.org/10.1109/CogInfoCom.2016.7804556. IEEE.

Cliburn, D. C. (2006): Experiences with the LEGO Mindstorms throughout the undergraduate computer science curriculum, 36th Annual Frontiers in Education Conference, 2006, pp. 1–6. IEEE

Csapó, B. (2003). A képességek fejlődése és iskolai fejlesztése. Akadémiai Kiadó, Budapest.

Csapó, G. (2017a). Sprego Virtual Collaboration Space. 8th IEEE International Conference on Cognitive Infocommunications, CogInfoCom, 2017, pp. 137–142. http://ieeexplore.ieee.org/document/8268230/. Letöltés dátuma: 2019.08.14. DOI=http://doi.org/10.1109/CogInfoCom.2017.8268230. IEEE.

Csapó, G. (2017b). Sprego Virtual Collaboration Space: Improvement Guidelines for the MaxWhere Seminar System. 8th IEEE International Conference on Cognitive Infocommunications, CogInfoCom, 2017, pp. 143–144. http://ieeexplore.ieee.org/document/8268231/. Letöltés dátuma: 2019.08.14. DOI=http://doi.org/10.1109/CogInfoCom.2017.8268231. IEEE.

Csapó, G. (2017c). Az informatika érettségi és az informatikushallgatók szövegkezelési kultúrája. Válogatott tanulmányok a pedagógiai elmélet és szakmódszertanok köréből. http://www.irisro.org/pedagogia2017januar/85CsapoGabor.pdf Letöltés dátuma: 2019.09.11.

Csapó, G. (2019). Placing event-action based visual programming in the process of computer science education. Acta Polytechnica Hungarica, 2019, 16(2), Cognitive Infocommunications, pp. 35–57. http://www.uni-obuda.hu/journal/Csapo_89.pdf. Letöltés dátuma: 2019.08.14. DOI=http://doi.org/10.12700/APH.16.2.2019.2.3.

Csapó, G. & Sebestyén, K. (2017). Educational Software for the Sprego Method. The Turkish Online Journal of Educational Technology, INTE 2017 October, pp. 986–999. http://www.tojet.net/special/2017_10_1.pdf. Letöltés dátuma: 2019.08.14.

Csapó G., Csernoch M. & Abari K. (2019). Sprego: Case Study on the Effectiveness of Teaching Spreadsheet Management with Schema Construction. Education and Information Technologies. Accepted.

Csernoch, M. (1997). Methodological Questions of Teaching Word Processing. 3rd International Conference on Applied Informatics, Eger-Noszvaj, Hungary, pp. 375–382.

Csernoch, M. (2009). Teaching word processing – the theory behind. Teaching Mathematics and Computer Science, 2009, 1, pp. 119–137.

Csernoch, M. (2010). Teaching word processing – the practice. Teaching Mathematics and Computer Science, 2010, 8(2), pp. 247–262.

Csernoch, M. (2011). Clearing Up Misconceptions About Teaching Text Editing. 4th International Conference of Education, Research and Innovation, ICERI2011, Madrid, Spain, pp. 407–415.

Csernoch, M. (2014). Programozás táblázatkezelő függvényekkel – Sprego. Műszaki Könyvkiadó, Budapest.

Csernoch, M. (2016). Algoritmusok és sémák az informatika oktatásában II. http://tanarkepzes.unideb.hu/szaktarnet/kiadvanyok/algoritmusok_es_semak_2.pdf. Letöltés dátuma: 2016.01.25.

Csernoch, M. (2017). Thinking Fast and Slow in Computer Problem Solving. Journal of Software Engineering and Applications, 2017, 10(1). http://file.scirp.org/pdf/JSEA_2017012315324696.pdf. Letöltés dátuma: 2017.07.08.

Csernoch, M. (2019a). Do You Speak and Write in Informatics? The 10th International Multi-Conference on Complexity, Informatics and Cybernetics, Orlando, Florida, USA.

Csernoch, M. (2019b). Knowledge Transfer in End-User Computing. The 10th International Multi-Conference on Complexity, Informatics and Cybernetics, Orlando, Florida, USA.

Csernoch, M. & Biró, P. (2014a). Spreadsheet misconceptions, spreadsheet errors. Oktatáskutatás határon innen and túl. HERA Évkönyvek I. Belve-dere Meridionale, Szeged, pp. 370–395.

Csernoch, M. & Biró, P. (2014b). Sprego programming. Spreadsheets in Education (eJSiE), 2014, 8(1). http://epublications.bond.edu.au/cgi/viewcontent.cgi?article=1175&context=ejsie. Letöltés dátuma: 2015.04.03.

Csernoch, M. & Biró, P. (2015a). Számítógépes problémamegoldás. TMT, Tudományos és Műszaki Tájékoztatás, Könyvtár- és információtudományi szakfolyóirat, 2015, 62(3), pp. 86–94.

Csernoch, M. & Biró, P. (2015b). Wasting Human and Computer Resources. International Journal of Social, Education, Economics and Management Engineering, 2015, 9(2), pp. 573–581.

Csernoch, M. & Biró, P. (2016). Teaching methods are erroneous: approaches which lead to erroneous end-user computing. European Spreadsheet Risk Interest Group Conference, EuSpRIG2016, 2016, London. http://www.eusprig.org/mcsernoch-2016.pdf. Letöltés dátuma: 2016.07.21.

Csernoch, M. & Biró, P. (2017). Sprego, End-user Programming in Spreadsheets. PPIG 2017, 28th Annual Workshop, 2017.07.01.–2017.07.03., Delft, Netherlands.

Csernoch, M. & Biró, P. (2018). Edu-Edition Spreadsheet Competency Framework. European Spreadsheet Risk Interest Group Conference, EuSpRIG2017, 2017, Imperial College, London, pp. 121–136.

Csernoch, M. & Bujdosó, Gy. (2009). Vizsga- és versenyfeladatok szövegbeviteli hibái és ezek következményei. Új Pedagógia Szemle, 2009, 1, pp. 19–40.

Csernoch, M. & Dani, E. (2017). Data-structure validator: an application of the HY-DE model. 8th IEEE International Conference on Cognitive Infocommunications, CogInfoCom, 2017, pp. 197–202. IEEE.

Csernoch, M., Biró, P., Abari, K. & Máth, J. (2014a). Programozásorientált táblázatkezelői függvények. XIV. Országos Neveléstudományi Konferencia, 2014.11.06.–2014.11.08., Debrecen, Hungary.

Csernoch, M., Simon, K., Brósch, É. & Kiss, É. (2014b). Spregoval tanultam táblázatkezelést. INFO Éra Konferencia, 2014, NJSZT, Zamárdi.

Csernoch, M., Biró, P., Máth, J. & Abari, K. (2015). Testing Algorithmic Skills in Traditional and Non-Traditional Programming Environments. Informatics in Education, 2015, 14(2), pp. 175–197. DOI=http://doi.org/10.15388/infedu.2015.11.

Dancsó, T. & Korom, P. (2013). Informatika 9-10. a gimnáziumok számára. Nemzedékek Tudása, Budapest.

ECDL Foundation (2016). ECDL Module | Spreadsheets: Syllabus Version 6.0. http://ecdl.org/media/ecdlspreadsheets_6.01.pdf. Letöltés dátuma: 2019.03.14.

ELTE IK T@T Labor (2019). Archivum | e-Hód. http://e-hod.elte.hu/archivum/. Letöltés dátuma: 2019.08.04.

Epic Games (2019). Blueprints Visual Scripting. https://docs.unrealengine.com/en-US/Engine/Blueprints/index.html. Letöltés dátuma: 2019.08.13.

EuSpRIG, European Spreadsheet Risk Interest Group (2019). EuSpRIG Horror Stories. http://www.eusprig.org/horror-stories.htm. Letöltés dátuma: 2019.06.28.

Fesakis, G. & Serafeim, K. (2009). Influence of the familiarization with scratch on future teachers' opinions and attitudes about programming and ICT in education. ACM SIGCSE Bulletin, ACM, 2009, 41(3), pp. 258–262.

Fincher, S., Cooper, S., Kölling, M. & Maloney, J. (2010). Comparing alice, greenfoot & scratch. 41st ACM technical symposium on Computer science education, ACM, 2010, pp. 192–193.

Fowler, A., Fristce, T. & MacLauren, M. (2012). Kodu Game Lab: a programming environment. The Computer Games Journal, 2012, 1(1), pp. 17–28.

Franklin, D., Hill, C., Dwyer, H. A., Hansen, A. K., Iveland, A. & Harlow, D. B. (2016). Initialization in scratch: Seeking knowledge transfer. 47th ACM Technical Symposium on Computing Science Education, ACM, 2016, pp. 217–222.

Galambos, P., Fülöp, I. M. & Baranyi, P. (2011). Virtual collaboration arena, platform for research, development and education. Acta Technica Jaurinensis, 2011, 4(1), pp. 145–155.

Garrett, N. (2015). Textbooks for Responsible Data Analysis in Excel. Journal of Education for Business, 2015, 90(4), pp. 169-174. DOI=http://doi.org/10.1080/08832323.2015.1007908.

Gilányi, A. & Virágos, M. (2013). Library treasures in a virtual world. 4th IEEE International Conference on Cognitive Infocommunications, CogInfoCom, 2013, pp. 563–566. IEEE.

Gilányi, A., Bálint, M., Museum, H., Hajdu, R., Tarsoly, S. & Erdős, I. (2015a). A visualization of the medieval Church of Zelemér. 6th IEEE International Conference on Cognitive Infocommunications, CogInfoCom, 2015, pp. 449–453. IEEE. DOI=http://doi.org/10.1109/CogInfoCom.2015.7390635.

Gilányi, A., Bálint, M., Hajdu, R., Tarsoly, S. & Erdős, I. (2015b). Presentation of the Church of Zelemér in the Virtual Collaboration Arena (VirCA). 6th IEEE International Conference on Cognitive Infocommunications, CogInfoCom, 2015. IEEE. DOI=http://doi.org/10.1109/CogInfoCom.2015.7390658.

Gould, J. (1975). Some psychological evidence on how people debug computer programs. International Journal of Man-Machine Studies, 1975, (7)1, pp. 151–182.

Gould, J. & Drongowski, P. (1974). An exploratory study of computer program debugging. Human Factors, 1974, 16(3), pp. 258–277.

Gulácsi, Á. & Dienes, N. (2018). 3D Software Environment for Educational Sprego Programming. The Turkish Online Journal of Educational Technology, INTE 2018 November (2), pp. 837–844. http://www.tojet.net/special/2018_12_3.pdf. Letöltés dátuma: 2019.01.21.

Hermans, F. (2019). How to teach programming (and other things)?. Strange Loop, 2019, St. Louis, MO. https://www.youtube.com/watch?v=g1ib43q3uXQ. Letöltés dátuma: 2019.11.18.

Horváth, I. (2018a). Evolution of teaching roles and tasks in VR / AR-based education, 9th IEEE International Conference on Cognitive Infocommunications, CogInfoCom, 2018, pp 355–360. DOI=http://doi.org/10.1109/CogInfoCom.2018.8639907. IEEE.

Horváth, Zs. T. (2018b). Another e-learning method in upper primary school: 3D spaces. 9th IEEE International Conference on Cognitive Infocommunications, CogInfoCom, 2018, pp. 405–408. DOI=http://doi.org/10.1109/CogInfoCom.2018.8639937. IEEE.

Hromkovič, J. (2009). Algorithmic Adventures. Springer, Heidelberg, Berlin.

Hubwieser, P. (2004). Functional Modelling in Secondary Schools Using Spreadsheets. Education and Information Technologies, 2004, 9(2), pp. 175–183. DOI=http://doi.org/10.1023/B:EAIT.0000027929.91773.ab.

Iacob, I. (2009). The effectiveness of computer assisted classes for English as a second language. arXiv preprint. https://arxiv.org/ftp/arxiv/papers/0905/0905.4611.pdf. Letöltés dátuma: 2017.07.11.

ICAEW (2016). Spreadsheet competency framework: A structure for classifying spreadsheet ability in finance professionals. ICAEW, London. http://www.icaew.com/-/media/corporate/files/technical/information-technology/it-faculty/spreadsheet-competency-framework.ashx. Letöltés dátuma: 2019.08.14.

Jerinic, L. (2014). Teaching Introductory Programming Agent-based Approach with Pedagogical Patterns for Learning by Mistake. International Journal of Advanced Computer Science and Applications, 2014, (5)6.

Kahneman, D. (2011). Thinking, Fast and Slow. Farrar, Straus and Giroux, New York.

Kalelioglu, F. & Gülbahar, Y. (2014). The effects of teaching programming via Scratch on problem solving skills: a discussion from learners' perspective. Informatics in Education, 2014, 13(1), pp. 33–50.

Kátai, Z., Tóth, L. & Adorjáni, A. K. (2014). Multi-Sensory Informatics Education. Informatics in Education, 2014, 13(2), pp. 225–240.

Kátai, Z., Osztián, E. & Vekov, G. K. (2016). Promoting computational thinking by artistically enhanced algorithm visualization. INFODIDACT 2015 Informatika Szakmódszertani Konferencia, 2016, Zamárdi, pp. 1–12. https://people.inf.elte.hu/szlavi/InfoDidact16/Manuscripts/KZOEVGK.pdf. Letöltés dátuma: 2019.08.14.

Katz, A. (2010). Beginning Microsoft Excel 2010. Apress. DOI=http://doi.org/10.1007/978-1-4302-2956-8.

Kecskés, I. (1987). Mikroszámítógépek használata az idegennyelv-oktatásában. Tankönyvkiadó Vállalat, Budapest.

Kim, H., Choi, H., Han, J. & So, H. J. (2012). Enhancing teachers' ICT capacity for the 21st century learning environment: Three cases of teacher education in Korea. Australasian Journal of Educational Technology. 2012, 28(6).

Klassner, F. & Anderson, S. D. (2003). Lego MindStorms: Not just for K-12 anymore, IEEE Robotics & Automation Magazine, 2003, 10(2), pp. 12–18.

Kruck, S. E., Maher, J. J. & Barkhi R. (2003). Framework for Cognitive Skill Acquisition and Spreadsheet Training. Journal of Organizational and End User Computing, 2003, 15(1), pp. 20–37.

Kruger, J. & Dunning, D. (1999). Unskilled and Unaware of It: How Difficulties in Recognizing One's Own Incompetence Lead to Inflated Self-Assessments. Journal of Personality and Social Psychology, 1999, 77(6), pp. 1121.

Lampert, B., Pongracz, A., Sipos, J., Vehrer, A. & Horvath, I. (2018). MaxWhere VR-Learning Improves Effectiveness over Clasiccal Tools of e-learning. Acta Polytechnica Hungarica, 2018, 15(3), pp. 125–147. http://www.uni-obuda.hu/journal/Lampert_Pongracz_Sipos_Vehrer_Horvath_82.pdf. Letöltés dátuma: 2018.09.12.

Lister, R. (2016). Toward a Developmental Epistemology of Computer Programming, 11th Workshop in Primary and Secondary Computing Education, ACM, 2016, pp. 5–16. http://www-staff.it.uts.edu.au/~raymond/talks/wipsce2016.pptx. Letöltés dátuma: 2016.12.15.

Meerbaum-Salant, O., Armoni, M. & Ben-Ari, M. (2011). Habits of programming in scratch. 16th annual joint conference on Innovation and technology in computer science education, ACM, 2011, pp. 168–172.

Merriënboer, J.J.G. van & Sweller, J. (2005). Cognitive Load Theory and Complex Learning: Recent Developments and Future Directions. Educational Psychology Review, 2005, 17(2), pp. 147–177. DOI=http://doi.org/10.1007/s10648-005-3951-0.

Message, R. H. (2013). Programming for humans: a new paradigm for domain-specific languages. Doctoral dissertation, University of Cambridge. https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-843.pdf. Letöltés dátuma: 2016.07.21.

Microsoft (2016). Microsoft Official Academic Course Microsoft Excel 2016: Includes coverage of the following Microsoft Office Specialist (MOS) exam: MOS EXAM 77-727: EXCEL 2016. John Wiley and Sons. https://www.dit.ie/media/ittraining/msoffice/MOAC_Excel_2016_Core.pdf. Letöltés dátuma: 2019.03.14.

Moyo, M., Tiba, A. C. & Madzima, K. (2016). Impact of Pre-Service Teachers' Prior Knowledge of Information Technologies on Perceptions and Beliefs on Computers in Education Modules. http://uir.unisa.ac.za/bitstream/handle/10500/22888/Moses%20Moyo%2C%20Anyen%20Chantylee%20Tiba%2C%20Kudakwashe%20Madzima.pdf?sequence=1&isAllowed=y. Letöltés dátuma: 2019.02.03.

National Research Council (2000). How people learn: Brain, mind, experience, & school: Expanded edition. National Academies Press, Washington.

OFI (2012). Nemzeti alaptanterv. Oktatáskutató és Fejlesztő Intézet, Budapest. http://ofi.hu/sites/default/files/attachments/mk_nat_20121.pdf. Letöltés dátuma: 2016.10.22.

OFI (2013). Kerettanterv a gimnáziumok 9-12. évfolyama számára. Oktatáskutató és Fejlesztő Intézet, Budapest. http://kerettanterv.ofi.hu/03_melleklet_9-12/index_4_gimn.html. Letöltés dátuma: 2016.11.09.

Osztián, E., Kátai, Z. & Vekov, G. K. (2017). Multi-Dimensional Expansion of Algo-Rythmics. The Turkish Online Journal of Educational Technology, INTE 2017 November, pp. 573–578. http://www.tojet.net/special/2017_11_2.pdf. Letöltés dátuma: 2019.01.10.

Panko, R. (2013). The Cognitive Science of Spreadsheet Errors: Why Thinking is Bad. 46th Hawaii International Conference on System Sciences, Maui, 2013, pp. 4013–4022. IEEE.

Panko, R. (2016). What We Don’t Know About Spreadsheet Errors Today: The Facts, Why We Don't Believe Them, and What We Need to Do. arXiv preprint. http://arxiv.org/abs/1602.02601. Letöltés dátuma: 2016.07.21.

Panko, R. & Port, D. (2013). End User Computing: The Dark Matter (and Dark Energy) of Corporate It. Journal of Organizational and End User Computing, 2013, 25(3), pp. 1–19.

Papadakis, S., Kalogiannakis, M., Orfanakis, V. & Zaranis, N. (2014). Novice programming environments. Scratch & app inventor: a first comparison. Workshop on Interaction Design in Educational Environments, ACM, 2014.

Pólya, G. (1954). How To Solve It. A New Aspect of Mathematical Method. 2nd edition, 1957. Princeton University Press, Princeton, New Jersey.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B. & Kafai, Y. (2009). Scratch: programming for all, Communications of the ACM, 2009, 52(11), pp. 60–67.

Schneider, M. (2004). An Empirical Study of Introductory Lectures in Informatics Based on Fundamental Concepts. Informatics and Students Assessment, Lecture Notes in Informatics 1, pp. 123–133.

Schneider, M. (2005). A Strategy to Introduce Functional Data Modeling at School Informatics. R. T. Mittermeir, From Computer Literacy to Informatics Fundamentals, pp. 130–144. Springer, Berlin, Heidelberg, Germany. http://link.springer.com/chapter/10.1007/978-3-540-31958-0_16. Letöltés dátuma: 2017.07.10.

Sebestyén, K., Csapó, G. & Csernoch, M. (2018). Visualising Sprego Inequality Problems With 2D Representations. The Turkish Online Journal of Educational Technology, INTE 2018 November (2), pp. 888–898. http://www.tojet.net/special/2018_12_3.pdf. Letöltés dátuma: 2019.08.14.

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

Shams, L. & Seitz, A. R. (2008). Benefits of multisensory learning. Trends in cognitive sciences, 2008, 12(11), pp. 411–417.

Shulman, L. S. (1986). Those Who Understand: Knowledge Growth in Teaching. Educational Researcher, 1986, 15(2), pp. 4–14.

Shulman, L. S. (1987). Knowledge and Teaching. Foundations of the New Reform. Harvard Educational Review, 1987, 57(1), pp. 1–23.

Skemp, R. (1971). The Psychology of Learning Mathematics. Lawrence Erlbaum Associatives, New Jersey, USA.

Soloway, E. (1993). Should we teach students to program?. Communications of the ACM, 1993, 36(10), pp. 21–25.

Stolee, K. T. & Fristoe, T. (2011). Expressing computer science concepts through Kodu game lab. 42nd ACM technical symposium on Computer science education, ACM, 2011, pp. 99–104.

Sykes, E. R. (2007). Determining the effectiveness of the 3D Alice programming environment at the computer science I level. Journal of Educational Computing Research, 2007, 36(2), pp. 223–244.

Szlávi, P. & Zsakó, L. (2019). Informatika oktatása. https://people.inf.elte.hu/szlavi/TAMOP-2/EgybenGeneralva/index.html. Letöltés dátuma: 2019.07.22.

Test ECDL (2019). Exam simulation ECDL Module 4 | Spreadsheets. https://www.testecdl.com/module-4-spreadsheets.html. Letöltés dátuma: 2019.02.22.

Wakeling, D. (2007). Spreadsheet functional programming. Journal of Functional Programming, 2007, 17(1), pp. 131–143. Cambridge University Press.

Walkenbach, J. (2010). Excel 2010 Bible. Wiley Publishing, Indianapolis.

Walkenbach, J. & Wilcox, C. (2003). 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. Letöltés dátuma: 2014.08.17.

Wilcox, C. & Walkenbach, J. (2003). Introducing array formulas in Excel. http://office.microsoft.com/en-us/excel-help/introducing-array-formulas-in-excel-HA001087290.aspx. Letöltés dátuma: 2014.08.22.

Wilson, R., Majsterek, D. & Simmons, D. (1996). The effects of computer-assisted versus teacher-directed instruction on the multiplication performance of elementary students with learning disabilities. Journal of Learning Disabilities, 1996, 29(4), pp. 382–390.

Wing, J. M. (2006a). Computational thinking. Communications of the ACM, 2006, 49(3), pp. 33–35.

Wing, J. M. (2006b). Computational Thinking. https://www.microsoft.com/en-us/research/wp-content/uploads/2012/08/Jeannette_Wing.pdf. Letöltés dátuma: 2019.11.18.

Zsakó, L. (2006). Combinatorics – Competition – Excel. Teaching Mathematics and Computer Science, 2006, 4(2), pp. 427–435.

13. Forrásjegyzék

Alexander, J. (2016). Construct 2 - From Beginner to Advanced - Ultimate Course! | Udemy. https://www.udemy.com/construct-2-from-beginner-to-advanced-build-10-games. Letöltés dátuma: 2019.07.13.

Béres, I. (2006). Tabulátorok használata. https://www.krudy-szeged.hu//home/static/kruszam/oktatas/kit_feladat/word_5_11_1_fea.html. Letöltés dátuma: 2019.07.22.

Burcsi (2017). Scratch programozás. http://www.burcsi.hu/_inf/Scratch/. Letöltés dátuma: 2019.07.22.

Caffé Szamila Kft. (2017). Caffé Szamila Kft. Alapanyagok, kiegészítők Tea. http://www.szamila.hu/index.php?kat=5&Tea. Letöltés dátuma: 2019.07.08.

Carnegie Mellon University (2019). Alice - Tell Stories. Build Games. Learn to Program. https://www.alice.org. Letöltés dátuma: 2019.07.12.

Christo161 (2016). C++ programozás kezdőknek - változók, konstansok, literálok. https://itkezdoknek.blog.hu/2016/09/06/cpp_programozas_kezdoknek_valtozok_konstansok_literalok. Letöltés dátuma: 2019.07.22.

Csapó, G. & Sebestyén K. (2019). Sprego - A hivatalos Sprego alkalmazás. https://play.google.com/store/apps/details?id=hu.sprego.oktatoprogram. Letöltés dátuma: 2019.08.12.

Google (2019). Google Dokumentumok – hozzon létre és szerkesszen dokumentumokat ingyenesen az interneten. https://www.google.hu/intl/hu/docs/about/. Letöltés dátuma: 2019.08.12.

Hutong Games (2019). PlayMaker - Visual Scripting for Unity. https://hutonggames.com/. Letöltés dátuma: 2019.08.13.

JobTestPrep (2016). SAM (Software Asset Management) Intermediate/Advanced Excel Practice Tests for Interview – 2016. https://www.jobtestprep.com/excel-practice-test-advanced-2016. Letöltés dátuma: 2019.04.14.

Karakas, H. N. (2003). Mozirajongók. https://arato.inf.unideb.hu/csernoch.maria/tanarkepzes/mozirajongok.rtf. Letöltés dátuma: 2019.06.29.

Lego (2019). Home - LEGO.com. https://www.lego.com/en-gb/mindstorms?ignorereferer=true. Letöltés dátuma: 2019.07.12.

Lifelong Kindergarten Group (2019). Scratch - Imagine, Program, Share. https://scratch.mit.edu. Letöltés dátuma: 2019.07.12.

Mandula Família Kft. (2017). Kismandula Cukrászda. http://www.kismandula.hu/klasszikus-tortak/tortak-2. Letöltés dátuma: 2017.08.10.

Microsoft (2019a). Excel functions (alphabetical). https://support.office.com/en-us/article/excel-functions-alphabetical-b3944572-255d-4efb-bb96-c6d90033e188. Letöltés dátuma: 2019.07.09.

Microsoft (2019b). Office 365 Login | Microsoft Office. https://www.office.com/. Letöltés dátuma: 2019.08.12.

Microsoft Research (2019). Kodu | Home. Available: https://www.kodugamelab.com. Letöltés dátuma: 2019.07.12.

MISTEMS Ltd. (2019). MaxWhere Store - VR workspaces. http://www.maxwhere.com/. Letöltés dátuma: 2019.07.16.

Nytta (2019a). Vásárlás a piacon. http://info.nytta.hu/temak/excel/kep/v01piac.jpg. Letöltés dátuma: 2019.07.21.

Nytta (2019b). Word feladatok, Karakterformázás. http://nytta.faludinet.hu/temak/word/mintak/f01_f7.gif. Letöltés dátuma: 2019.07.22.

OFI (2004). [Countries], Informatika érettségi 2004. https://www.oktatas.hu/bin/content/dload/erettsegi/probaerettsegi_2004/info_em_flap.zip. Letöltés dátuma: 2017.06.23.

Pohl, W. & Dagienė, V. (2019). What is Bebras | www.bebras.org. https://www.bebras.org/. Letöltés dátuma: 2019.08.04.

Scirra (2019a). Game Making Software - Construct 3. https://www.construct.net/en. Letöltés dátuma: 2019.07.13.

Scirra (2019b). Welcome to the Construct 3 Manual. https://www.construct.net/en/make-games/manuals/construct-3. Letöltés dátuma: 2019.08.13.

Scirra (2019c). Game Development Tutorials. https://www.construct.net/en/tutorials?flang=1. Letöltés dátuma: 2019.08.13.

Scirra (2019d). Construct Forum. https://www.construct.net/en/forum. Letöltés dátuma: 2019.08.13.

ScirraVideos (2019). ScirraVideos - YouTube. https://www.youtube.com/user/ScirraVideos. Letöltés dátuma: 2019.07.13.

Sulinet (2019). Egyszerű számítások a táblázatkezelő programban. https://tudasbazis.sulinet.hu/hu/szakkepzes/ugyvitel/informaciokezeles/egyszeru-szamitasok-a-tablazatkezelo-programban/az-elemi-matematikai-fuggvenyek. Letöltés dátuma: 2019.07.21.

Száva, M. (2010). Évzáró sakkhét DEBRECENBEN. http://video.dehir.hu/dokumentumok/3az1ben_sakkversenyek.doc. Letöltés dátuma: 2019.07.22.

The Document Foundation (2019). Home | LibreOffice - Free Office Suite - Fun Project - Fantastic People. https://www.libreoffice.org/. Letöltés dátuma: 2019.08.12.

The R Foundation (2019). The R Project for Statistical Computing. https://www.r-project.org/. Letöltés dátuma: 2019.08.04.

Tóth, R. (2017). StreamWriter/Kiiras. http://users.atw.hu/toth23/StreamWriter.JPG. Letöltés dátuma: 2019.07.22.

UNESCO World Heritage Centre (2019). Worldb Heritage List. http://whc.unesco.org/?cid=31&mode=table. Letöltés dátuma: 2019.07.23.

YoYo Games (2019). Drag And Drop Overview. https://docs2.yoyogames.com/. Letöltés dátuma: 2019.08.13.

14. Publikációs jegyzék

Magyar nyelvű könyvrészletek

Csapó, G. (2017). Az informatika érettségi és az informatikushallgatók szövegkezelési kultúrája. Válogatott tanulmányok a pedagógiai elmélet és szakmódszertanok köréből. V. Neveléstudományi és Szakmódszertani Konferencia, 2017, Párkány. http://www.irisro.org/pedagogia2017januar/85CsapoGabor.pdf. Letöltés dátuma: 2019.08.14. DOI=http://doi.org/10.18427/iri-2017-0053.

Idegen nyelvű tudományos közlemények nemzetközi folyóiratban

Csapó, G., Csernoch, M. & Abari, K. (2019). Sprego: Case Study on the Effectiveness of Teaching Spreadsheet Management with Schema Construction. Education and Information Technologies, 2019, pp. 1-21. https://link.springer.com/article/10.1007/s10639-019-10024-2. DOI=10.1007/s10639-019-10024-2.

Csapó, G. (2019). Placing event-action based visual programming in the process of computer science education. Acta Polytechnica Hungarica, 2019, 16(2), Cognitive Infocommunications, pp. 35–57. http://www.uni-obuda.hu/journal/Csapo_89.pdf. Letöltés dátuma: 2019.08.14. DOI=http://doi.org/10.12700/APH.16.2.2019.2.3.

Idegen nyelvű tudományos konferencia közlemények

Sebestyén, K., Csapó, G. & Csernoch, M. (2018). Visualising Sprego Inequality Problems With 2D Representations. The Turkish Online Journal of Educational Technology, INTE 2018 November (2), pp. 888–898. http://www.tojet.net/special/2018_12_3.pdf. Letöltés dátuma: 2019.08.14.

Csapó, G. & Sebestyén, K. (2017). Educational Software for the Sprego Method. The Turkish Online Journal of Educational Technology, INTE 2017 October, pp. 986–999. http://www.tojet.net/special/2017_10_1.pdf. Letöltés dátuma: 2019.08.14.

Csapó, G. (2017). Sprego Virtual Collaboration Space. 8th IEEE International Conference on Cognitive Infocommunications, CogInfoCom, 2017, pp. 137–142. http://ieeexplore.ieee.org/document/8268230/. Letöltés dátuma: 2019.08.14. DOI=http://doi.org/10.1109/CogInfoCom.2017.8268230. IEEE

Csapó, G. (2017). Sprego Virtual Collaboration Space: Improvement Guidelines for the MaxWhere Seminar System. 8th IEEE International Conference on Cognitive Infocommunications, CogInfoCom, 2017, pp. 143–144. http://ieeexplore.ieee.org/document/8268231/. Letöltés dátuma: 2019.08.14. DOI=http://doi.org/10.1109/CogInfoCom.2017.8268231. IEEE

Magyar nyelvű tudományos konferencia közlemények

Csapó, G. & Sebestyén, K. (2015). Oktatóprogram a Sprego táblázatkezelő módszerhez. INFODIDACT 2015 Informatika Szakmódszertani Konferencia, 2015, Zamárdi. http://people.inf.elte.hu/szlavi/InfoDidact15/Manuscripts/CsGSK.pdf. Letöltés dátuma: 2019.08.14.

Csapó, G. (2014). Vizuális programozás oktatása középiskolákban. INFODIDACT 2014 Informatika Szakmódszertani Konferencia, 2014, Zamárdi. http://people.inf.elte.hu/szlavi/InfoDidact14/Manuscripts/CsG.pdf Letöltés dátuma: 2019.08.14.

Beküldött, közlésre elfogadott tudományos közlemények

Sebestyén, K., Csapó, G., Csernoch, M., & Aradi, B. (2019). Error Recognition Model: End-user Text Management. Informatics in Education. Submitted.

Csapó, G. & Sebestyén, K. (2019). Bevezetés az algoritmus alapú adatkezelésbe táblázatkezelői környezetben: Digitális Témahét tapasztalatai. ADA 2019 konferencia, 2019, Debrecen. Accepted.

Sebestyén, K., Csapó, G. & Csernoch, M. (2019). Introduction to Algorithmic Based Data Management in Spreadsheet Environment. The Turkish Online Journal of Educational Technology. Accepted.

Szakmai konferenciák, előadástartások idegen nyelven

Időpont	Előadás adatai
2019. július	International Conference on New Horizons in Education nemzetközi tudományos konferencia: *Introduction to algorithmic based data management in spreadsheet environment*, Prága
2018. július	International Conference on New Horizons in Education nemzetközi tudományos konferencia: *Visualising Sprego inequality problems with 2D representations*, Párizs
2017. szeptember	8th IEEE International Conference on Cognitive Infocommunications nemzetközi tudományos konferencia: *Sprego Virtual Collaboration Space*, Debrecen
2017. szeptember	8th IEEE International Conference on Cognitive Infocommunications nemzetközi tudományos konferencia: *Sprego Virtual Collaboration Space: Improvement Guidelines for the MaxWhere Seminar System*, Debrecen
2017. július	International Conference on New Horizons in Education nemzetközi tudományos konferencia: *Educational Software for the Sprego Method*, Berlin

Szakmai konferenciák, előadástartások magyar nyelven

Időpont	Előadás adatai
2019. május	ADA 2019 tudományos konferencia: *Bevezetés az algoritmus alapú adatkezelésbe táblázatkezelői környezetben: Digitális Témahét tapasztalatai*, Debrecen
2019. április	Digitális Témahét: *Informatika didaktika módszertani bemutató szakórák*, Szlovák Tanítási Nyelvű Óvoda, Általános Iskola, Gimnázium és Diákotthon, Budapest
2018. november	Az informatikaoktatás szabályozása és módszertana a közoktatásban akkreditált pedagógus továbbképzési program: *A Sprego módszertant támogató 2D vizuális reprezentációk bővítése*, Zamárdi
2018. február	Oktatás – Informatika – Pedagógia 2018 tudományos konferencia: *A Sprego módszertan támogatása interaktív és virtuális valóságra alapuló eszközökkel*, Debrecen
2017. november	Az informatikaoktatás szabályozása és módszertana a közoktatásban akkreditált pedagógus továbbképzési program: *Komplex tanulási környezetek konstruálása virtuális térben*, Zamárdi

2017. augusztus	Az Informatika a felsőoktatásban 2017 konferencia: *Az esemény-utasítás alapú vizuális programozás helye az informatikaoktatás folyamatában,* Debrecen
2017. március	XXXIII. OTDK Konferencia Tanulás- és Tanításmódszertani – Tudástechnológiai Szekció, különdíj: *Sprego oktatószoftver*, Győr
2017. január	V. Neveléstudományi és Szakmódszertani Konferencia nevű nemzetközi tudományos konferencia: *Az informatika érettségi és az informatikushallgatók szövegkezelési kultúrája*, Párkány
2016. november	Az informatikaoktatás szabályozása és módszertana a közoktatásban akkreditált pedagógus továbbképzési program: *Mobil alkalmazás fejlesztő workshop*, Zamárdi
2016. november	Debreceni Egyetem Informatikai Kar Szakmai Napok: *Játékfejlesztés vizuálisan workshop,* Debrecen
2016. október	Europe Code Week 2016, Debreceni Ady Endre Gimnázium: *Játékfejlesztés egyszerűen workshop,* Debrecen
2016. április	Debreceni Egyetem Hatvani István Szakkollégium Tavaszi Tudományos Hallgatói Konferencia: *Tanulási segédanyag a Sprego táblázatkezelői módszerhez,* Debrecen
2016. április	XVII. Eötvös Konferencia: *Oktatóprogram a Sprego módszerhez,* Budapest
2016. március	Debreceni Egyetem Informatikai Kar Szakmai Napok: *Játékfejlesztés vizuálisan workshop,* Debrecen
2015. november	Az informatikaoktatás szabályozása és módszertana a közoktatásban akkreditált pedagógus továbbképzési program: *Oktatóprogram a Sprego táblázatkezelő módszerhez*, Zamárdi
2015. november	Debreceni Egyetem Informatikai Kar TDK Konferencia, 3. helyezés: *Sprego oktatóprogram,* Debrecen
2015. november	Debreceni Egyetem Informatikai Kar, Bevezetés az informatikába tanóra tanárszakos hallgatóknak: *Oktatószoftverek fejlesztése vizuális programozási eszközökkel,* Debrecen
2015. október	Debreceni Egyetem Informatikai Kar Szakmai Napok: *Scirra workshop,* Debrecen
2015. május	„Játékkal a világ körül" nemzetközi tudományos konferencia: *Vizuális programozás oktatása középiskolákban*, Hajdúböszörmény
2015. április	XXXII. OTDK Konferencia Tanulás- és Tanításmódszertani – Tudástechnológiai Szekció: *Vizuális programozás oktatása középiskolákban*, Sárospatak
2014. november	Debreceni Egyetem Informatikai Kar TDK Konferencia, 1. helyezés: *Vizuális programozás oktatása középiskolákban,* Debrecen
2014. november	Az informatikaoktatás szabályozása és módszertana a közoktatásban akkreditált pedagógus továbbképzési program: *Vizuális programozás oktatása középiskolákban*, Zamárdi
2014. április	XV. Eötvös Konferencia: *Vizuális programozás oktatása középiskolákban,* Budapest
2014. április	Debreceni Egyetem Informatikai Kar Lányok Napja: *Fejlessz játékot!,* Debrecen
2014. március	Debreceni Egyetem Informatikai Kar Szakmai Napok: *Multiplayer alkalmazások fejlesztése Construct 2 környezetben,* Debrecen

2013. november	Debreceni Egyetem Informatikai Kar TDK Konferencia, 1. helyezés: *Vizuális programozás a középiskolai oktatásban*, Debrecen
2013. október	Debreceni Egyetem Informatikai Kar Szakmai Napok: *A vizuális programozás hatékonysága*, Debrecen
2013. április	Debreceni Egyetem Informatikai Kar Lányok Napja: *Scirra,* Debrecen
2012. május	Debreceni Egyetem Informatikai Kar TDK Konferencia, 2. helyezés: *Klikk és Klavia Túra - Terminológiai következetlenségek és azok hatása a leggyakoribb beviteli eszközök használata során*, Debrecen

Részvétel kutatási projektekben

Időtartam	Projekt neve
2018 –	Az informatika Kerettanterv hatásvizsgálata a közoktatásban
2011 – 2016	Testing Algorithmic and Application Skills
2011 – 2014	IKT kompetenciák szinkronizációs lehetőségeinek vizsgálata a közép- és felsőfokú oktatásban

Szakmaspecifikus produktumok

Időtartam	Produktum neve
2015 – 2019	Sprego 2D oktatóprogram fejlesztése
2017 – 2019	Sprego virtuális kollaborációs tér elkészítése, karbantartása és frissítése

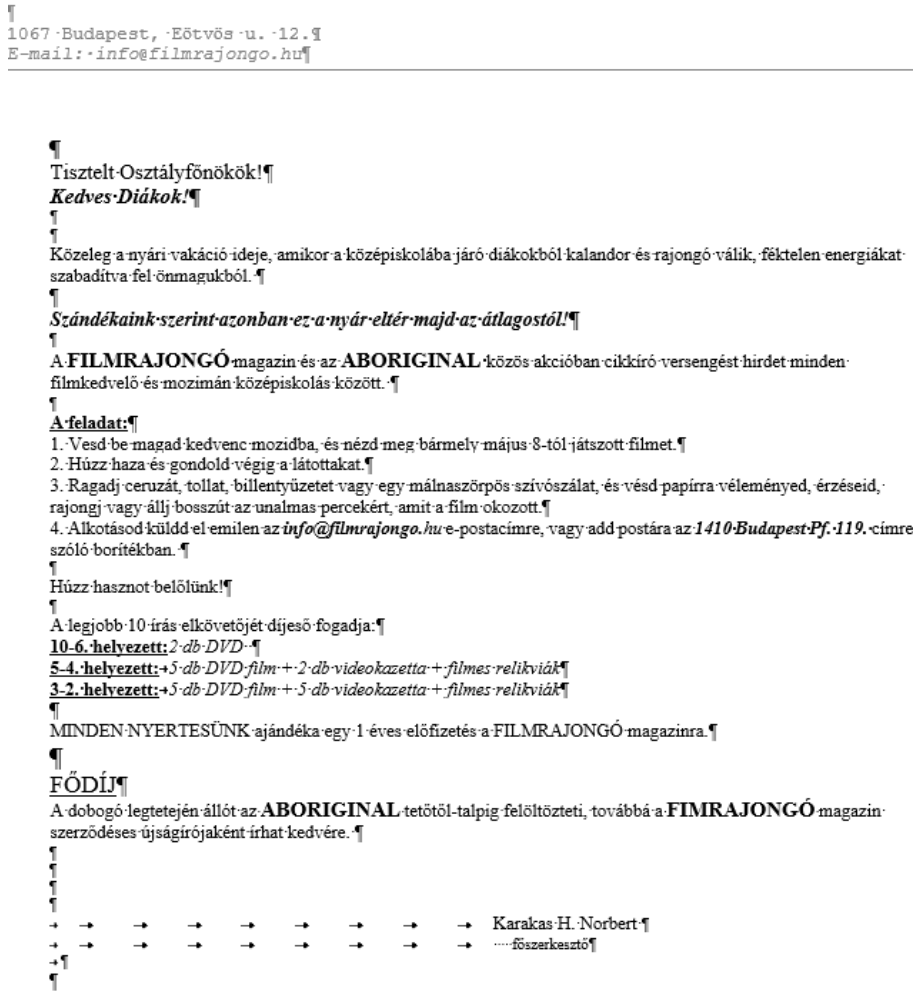
Részvétel szakmai továbbképzéseken

Időpont	Továbbképzés adatai
2019. november	*Az informatikaoktatás szabályozása és módszertana a közoktatásban 2019* című 30 órás akkreditált pedagógus továbbképzési program részvétel, Zamárdi
2019. június	*A pedagógusok digitális kompetenciájának megújítása. IKT eszközök rendszerszerű használata a tanulási-tanítási folyamatban – EFOP-3.2.4-16 – Alapozó továbbképzés* című 30 órás akkreditált pedagógus továbbképzési program tanúsítvány
2019. május	Pedagógus I. pedagógus besorolási fokozatra irányuló, 2019. évi általános eljárás minősítő vizsga tanúsítvány
2019. május	*Digitális szövegkezelés a hatékonyabb és biztonságosabb tanári és tanulói munka megvalósítása érdekében* című 30 órás akkreditált pedagógus továbbképzési program tanúsítvány
2018. november	*Az informatikaoktatás szabályozása és módszertana a közoktatásban 2018* című 30 órás akkreditált pedagógus továbbképzési program tanúsítvány, Zamárdi
2017. november	*Az informatikaoktatás szabályozása és módszertana a közoktatásban 2017* című 30 órás akkreditált pedagógus továbbképzési program tanúsítvány, Zamárdi
2017. szeptember	*Algoritmusok és sémák az informatika oktatásában* című 30 órás akkreditált pedagógus továbbképzési program tanúsítvány

2017. szeptember	*Felkészítés a köznevelési regisztrációs és tanulmányi alaprendszer (KRÉTA) pedagógus moduljaihoz* című 30 órás akkreditált pedagógus továbbképzési program tanúsítvány
2016. december	Certified Microsoft Innovative Educator tanúsítvány
2016. november	*Az informatikaoktatás szabályozása és módszertana a közoktatásban 2016* című 30 órás akkreditált pedagógus továbbképzési program tanúsítvány, Zamárdi
2015. november	*Az informatikaoktatás szabályozása és módszertana a közoktatásban 2015* című 30 órás akkreditált pedagógus továbbképzési program részvétel, Zamárdi
2014. november	*A Számítógépes gondolkodás fejlesztése az informatika modern eszközeivel* című szakmai továbbképzés részvétel, Debrecen
2014. november	*Az informatikaoktatás szabályozása és módszertana a közoktatásban 2014* című 30 órás akkreditált pedagógus továbbképzési program tanúsítvány, Zamárdi

15. Függelék

15.1. Példa egy hibás szöveges dokumentumra (Karakas, 2003)



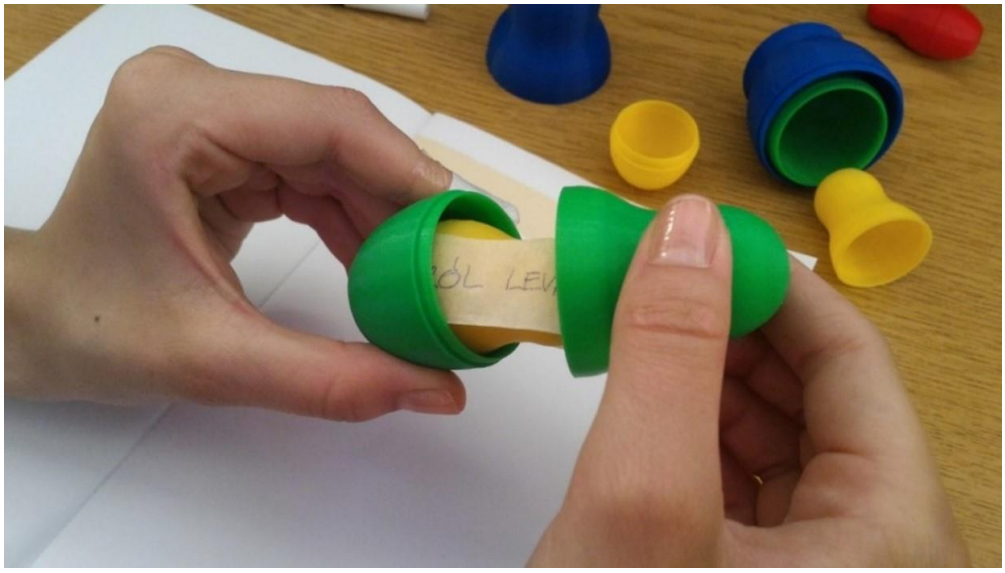
¶
1067 ·Budapest, ·Eötvös ·u. ·12.¶
E-mail: ·info@filmrajongo.hu¶

¶
Tisztelt·Osztályfönökök!¶
Kedves·Diákok!¶
¶
¶
Közeleg·a·nyári·vakáció·ideje, ·amikor·a·középiskolába·járó·diákokból·kalandor·és·rajongó·válik, ·féktelen·energiákat· szabadítva·fel·önmagukból.·¶
¶
Szándékaink·szerint·azonban·ez·a·nyár·eltér·majd·az·átlagostól!¶
¶
A·**FILMRAJONGÓ**·magazin·és·az·**ABORIGINAL**·közös·akcióban·cikkíró·versengést·hirdet·minden· filmkedvelő·és·mozimán·középiskolás·között.·¶
¶
A·feladat:¶
1.·Vesd·be·magad·kedvenc·mozidba, ·és·nézd·meg·bármely·május·8-tól·játszott·filmet.¶
2.·Húzz·haza·és·gondold·végig·a·látottakat.¶
3.·Ragadj·ceruzát, ·tollat, ·billentyűzetet·vagy·egy·málnaszörpös·szívószálat, ·és·vésd·papírra·véleményed, ·érzéseid, · rajongj·vagy·állj·bosszút·az·unalmas·percekért, ·amit·a·film·okozott.¶
4.·Alkotásod·küldd·el·emilen·az·***info@filmrajongo.hu***·e-postacímre, ·vagy·add·postára·az·***1410·Budapest·Pf.·119.***·címre szóló·borítékban.·¶
¶
Húzz·hasznot·belőlünk!¶
¶
A·legjobb·10·írás·elkövetőjét·díjeső·fogadja:¶
10-6.·helyezett:*2·db·DVD*·¶
5-4.·helyezett:→*5·db·DVD·film·+·2·db·videokazetta·+·filmes·relikviák*¶
3-2.·helyezett:→*5·db·DVD·film·+·5·db·videokazetta·+·filmes·relikviák*¶
¶
MINDEN·NYERTESÜNK·ajándéka·egy·1·éves·előfizetés·a·FILMRAJONGÓ·magazinra.¶
¶
FŐDÍJ¶
A·dobogó·legtetején·állót·az·**ABORIGINAL**·tetőtől-talpig·felöltözteti, ·továbbá·a·**FIMRAJONGÓ**·magazin· szerződéses·újságírójaként·írhat·kedvére.·¶
¶
¶
¶
¶
→ → → → → → → → → Karakas·H.·Norbert·¶
→ → → → → → → → → ····főszerkesztő¶
→¶
¶

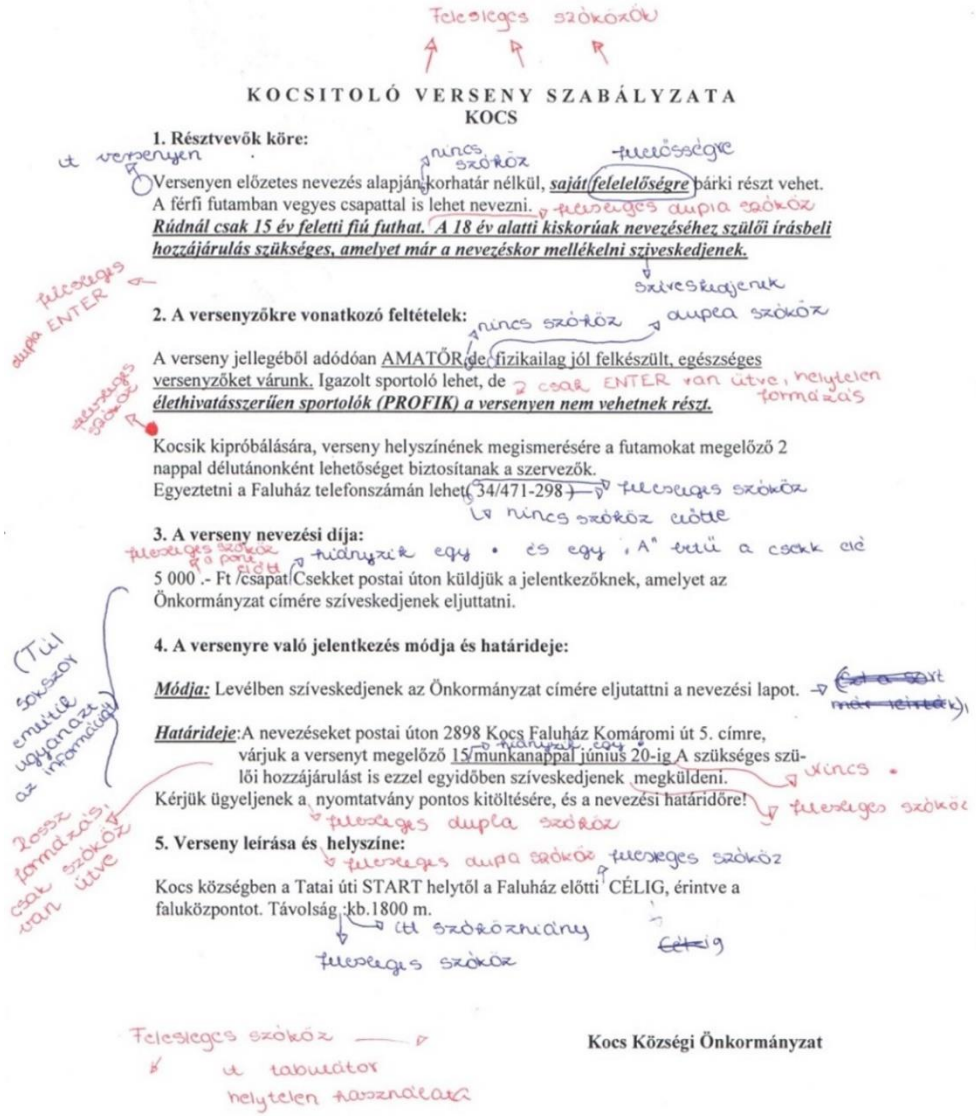
15.2. Példa egy hibás táblázatra (Caffé Szamila Kft., 2017)

	A	B	C
1	Rooibos tea vaníliával	627	100 g / csomag
2	Fekete tea / narancs - fahéj - vanília KG544/100	736	100 g ömlesztett szálas tea
3	Fekete tea / 'Török alma' / KG719/100	747	100 g ömlesztett szálas tea
4	LA VIA DEL TÉ CUKOR / BARNA / APRÓ	783	200 g
5	Menta tea 'Marrakech' KG735/100	799	100 g ömlesztett szálas tea
6	LA VIA DEL TÉ adagoló kanál szálas teához	810	
7	Hibiszkusz tea CV21	810	100 g ömlesztett szálas tea
8	Fekete tea L'Ombra del Vento KG748/100	820	
9	Zöld tea 'Special Gunpowder' KG G604/100	821	100 g ömlesztett szálas tea
71	Fekete tea citromhéj darabokkal KG318	8248	1000 g ömlesztett szálas tea
72	Fehér tea / Pai Mu Tan / KGW903	8370	500 g ömlesztett szálas tea
73	Fekete tea Ceylon OP1 KGHC1	8370	1000 g ömlesztett szálas tea
74	Gyümölcs tea / Crema al Whiskey KG613	8424	1000 g ömlesztett szálas tea
75	Gyümölcs tea / 'Giardini Naxos' KG626	8424	
76	Gyümölcs tea / Pinacolada KG501	8748	1000 g ömlesztett szálas tea
77	Gyümölcs tea / Sogno d'Amore - Szerelem álma KG568	9288	1000 g ömlesztett gyümölcs tea
78	Gyümölcs tea / Paradiso tea / KG567	9288	1000 g ömlesztett szálas tea
79	Fekete tea Earl Grey Imperiale KG411	9483	1000 g ömlesztett szálas tea
80	Fekete tea Assam TGFOP KGAS2	9990	1000 g ömlesztett szálas tea
81	Gyógytea "Kreativitás" KG846	9990	1000 g ömlesztett szálas tea
82	Zöld tea 'Bancha' cseresznyés KGJ383	13068	1000 g ömlesztett szálas tea
83	SELYEM TEAFILTER DÍSZDOBOZBAN	14850	
84	TEÁS KÍNÁLÓ (fémdobozok nélkül)	19710	
85	Szamovár 6L 'Rebecca' ezüst színű	115542	

15.3. Unplugged eszközök a Sprego módszertanhoz



15.4. Példa unplugged dokumentumra az ERM módszertanhoz



Felesleges szóközök

KOCSITOLÓ VERSENY SZABÁLYZATA
KOCS

1. Résztvevők köre:

Versenyen előzetes nevezés alapján, korhatár nélkül, ***saját felelősségre*** párki részt vehet.
A férfi futamban vegyes csapattal is lehet nevezni.
Rúdnál csak 15 év feletti fiú futhat. A 18 év alatti kiskorúak nevezéséhez szülői írásbeli hozzájárulás szükséges, amelyet már a nevezéskor mellékelni szíveskedjenek.

2. A versenyzőkre vonatkozó feltételek:

A verseny jellegéből adódóan AMATŐR, de fizikailag jól felkészült, egészséges versenyzőket várunk. Igazolt sportoló lehet, de ***élethivatásszerűen sportolók (PROFIK) a versenyen nem vehetnek részt.***

Kocsik kipróbálására, verseny helyszínének megismerésére a futamokat megelőző 2 nappal délutánonként lehetőséget biztosítanak a szervezők.
Egyeztetni a Faluház telefonszámán lehet: 34/471-298

3. A verseny nevezési díja:

5 000 .- Ft /csapat Csekket postai úton küldjük a jelentkezőknek, amelyet az Önkormányzat címére szíveskedjenek eljuttatni.

4. A versenyre való jelentkezés módja és határideje:

Módja: Levélben szíveskedjenek az Önkormányzat címére eljuttatni a nevezési lapot.

Határideje:A nevezéseket postai úton 2898 Kocs Faluház Komáromi út 5. címre, várjuk a versenyt megelőző 15/munkanappal június 20-ig.A szükséges szülői hozzájárulást is ezzel egyidőben szíveskedjenek megküldeni.
Kérjük ügyeljenek a nyomtatvány pontos kitöltésére, és a nevezési határidőre!

5. Verseny leírása és helyszíne:

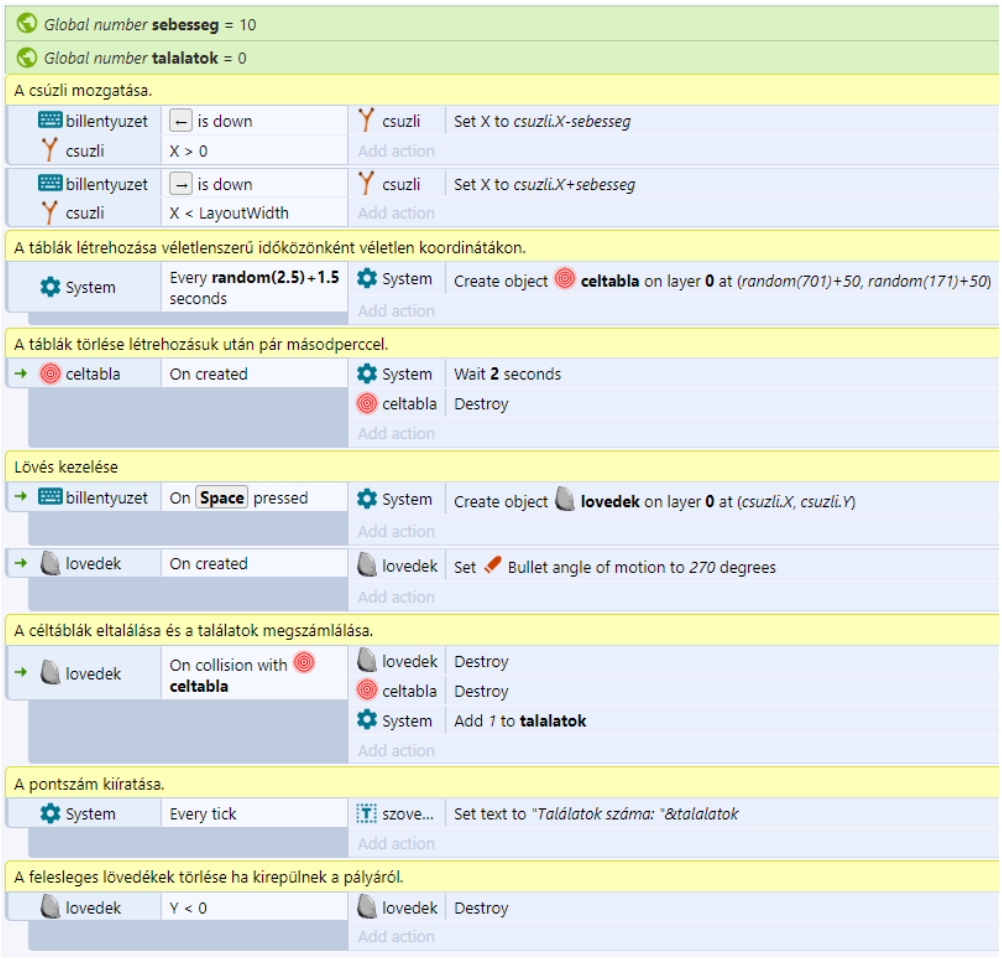
Kocs községben a Tatai úti START helytől a Faluház előtti CÉLIG, érintve a faluközpontot. Távolság :kb.1800 m.

Kocs Községi Önkormányzat

15.5. Sprego oktatóprogram: egyenlőtlenségen alapuló feltételes megszámlálás

15.6. Sprego oktatóprogram: lineáris keresés

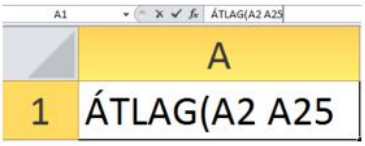
15.7. Példa összetett feladatra a Construct 3 környezetben



15.8. [T1] Adatgyűjtéshez használt tesztlap

Név: ..... Osztály: ..... Életkor.....
Dátum:.....

1. Hogyan egészítenéd ki az A1 cella tartalmát, hogy helyes képleted legyen? Írd a megoldásodat a pontozott vonalra!



.....

2. Kitaláltunk egy SPREGO() nevű függvényt. A függvénynek van két bemenő értéke: az első egy szöveg, a második egy egész szám. Karikázd be a helyesen írt képleteket!
   - SPREGO("AAAA";2)
   - SPREGO(aaaa;"2")
   - =SPREGO("AAAA";2)
   - =SPREGO(2;aaaa)
   - =SPREGO(B1;2)
   - =SPREGO(B1;C1+2)
   - =SPREGO("AAAA";2;5)
   - =SPREGO("B";3*2)

3. Hogyan hivatkozol az alábbi két cellatartományra?

	A	B	C	D	E
1	**Country**	**Continent**	**Capital**	**Area**	**Population (thousand)**
2	Afghanistan	Asia	Kabul	647500	27756
3	Albania	Europe	Tirana	28748	3545
4	Algeria	Africa	Algiers	2381740	32278
5	American Samo	Oceania	Pago Pago	199	69
6	Andorra	Europe	Andorra la Vella	468	68
7	Angola	Africa	Luanda	1246700	10593

Hivatkozás a világosszürke tartományra: .....

Hivatkozás a sötétszürke tartományra: .....

4. Írd le, hogy szerinted a táblázatkezelő milyen sorrendben hajtja végre az alábbi képletben szereplő műveleteket! Az 1. lépés mellé azt a műveletet írd, amelyet a táblázatkezelő először hajt végre! (A táblázat sorainak száma nem szükségszerűen egyezik meg a lépések számával.)

=INDEX(A2:A5;SPREGO(B1;MAX(D2:D5)-7))

1. lépés	
2. lépés	
3. lépés	
4. lépés	
5. lépés	
6. lépés	

5. Oldd meg az alábbi feladatokat táblázatkezelő képletekkel!

	A	B	C
1	Név - kód	Ár	
2	Áfonya Hercegnő cukormentes tortája - KT057	3950 Ft	XXXX
3	Őrség zöld aranya torta ország tortája 2016 - KT058	3950 Ft	
4	Az ország tortája 2015 - KT050	3950 Ft	
5	Oreo keksz torta - KT052	3950 Ft	
56	Málnácska torta - KT056	3800 Ft	
57	Lekváros kardinális torta - KT044	3500 Ft	
58	Karamellás csoki mousse torta - KT098	3800 Ft	

a. Írasd ki az összes tortának a kódját!

.....

b. Írasd ki az összes tortának a nevét!

.....

c. Az árak nem valódi számok, mert számjegyeket és betűket is tartalmaznak. Alakítsd át az összes árat valódi számmá!

.....

d. A C2 cellába begépeltünk egy árat. Írasd ki az összes tortánál, hogy **drága**, ha az ára nagyobb a C2-ben lévő árnál, minden más esetben írasd ki azt, hogy **olcsó**!

.....

e. A C2 cellába begépeltünk egy árat. Hány olyan torta van, amelyik drágább, mint a C2-ben lévő ár?

.....

f. A C2 cellába begépeltünk egy árat. Mennyibe kerülnek összesen azok a torták, amelyek olcsóbbak a C2-ben lévő árnál?

.....

g. A C2 cellába begépeltünk egy árat. Mennyi az átlagos ára azoknak a tortáknak, amelyek olcsóbbak a C2-ben lévő árnál?

.....

A kutatást az „Integrált kutatói utánpótlás-képzési program az informatika és számítástudomány diszciplináris területein” (EFOP-3.6.3-VEKOP-16-2017-00002) című projekt támogatta. A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósul meg.

15.9. [T2] Adatgyűjtéshez használt tesztlap

Név: ..... osztály: ..... dátum: .....

Oldja meg az a)–e) feladatokat táblázatkezelő képletekkel (MS Excel, OpenOffice, LibreOffice Calc), és mondja meg, hogy mit csinál az f) képlet!

	A	B	C	D	E	F	G
1	**ország**	**földrész**	**főváros**	**terület**	**lakosság (ezer)**		
2	Afghanistan	Asia	Kabul	647500	27756		
3	Albania	Europe	Tirana	28748	3545		
4	Algeria	Africa	Algiers	2381740	32278		
5	American Samoa	Oceania	Pago Pago	199	69		
6	Andorra	Europe	Andorra la Vella	468	68		
7	Angola	Africa	Luanda	1246700	10593		
8	Anguilla	Amerika	The Valley	102	12		
233	Yemen	Asia	Sanaa	527970	18701		
234	Yugoslavia	Europe	Belgrade	102350	10657		
235	Zambia	Africa	Lusaka	752614	9959		
236	Zimbabwe	Africa	Harare	390580	11377		

a) Írassa ki a legnagyobb területű ország fővárosát!

.....

b) Írassa ki az egyes országok népsűrűségét!

.....

c) Írassa ki az afrikai országok számát!

.....

d) Írassa ki azon országok átlagos területét, amelyek lakossága kisebb, mint G2!

.....

e) Írassa ki azon országok számát, amelyek területe nagyobb, mint G2!

.....

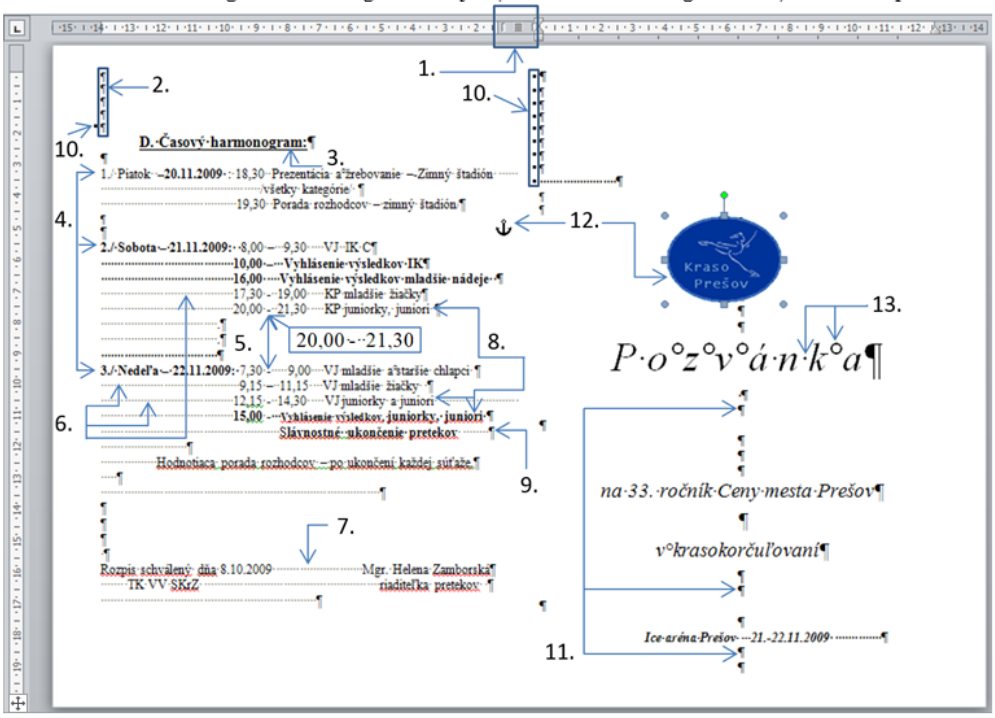
f) Mit csinál az alábbi összetett képlet?
{=SZUM(HA(B2:B236="Europe";HA(BAL(A2:A236)="A";1)))}

.....

15.10. [T3] Adatgyűjtéshez használt tesztlap

Név: ..... Életkor: ..... Neme: ☐ férfi ☐ nő

Értelmezze az alábbi szövegszerkesztési megoldásokat! Jelölje ×-szel a táblázat megfelelő cellájában a hiba típusát!



	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.
helyes													
tördelési hiba													
formázási hiba													
szintaktikai hiba													
szemantikai hiba													
tipográfiai hiba													

15.11. [T4] Adatgyűjtéshez használt tesztlap

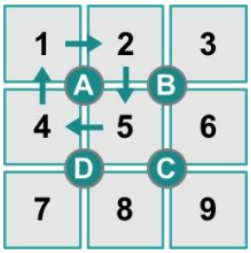
HÓDÍTSD MEG A BITEKET! – 2016-OS FELADATSOR KADÉT

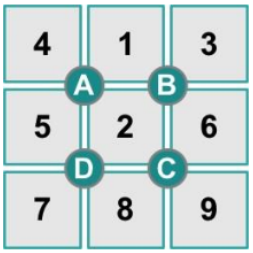
SZÁMFORDÍTÓ (2012-HU-01A)

A „számfordító” játékban a számokat 1-től 9-ig keverhetjük. A játék kezdetén a számok mindig a bal oldali képnek megfelelően vannak elhelyezve.

Ha az A, B, C vagy D gombok valamelyikét megnyomjuk, a gomb körül elhelyezkedő számok az óramutató járásával megegyező irányba fordulnak egyet.

Például az A gomb megnyomása után a számok a jobboldali képnek megfelelően helyezkednek el.





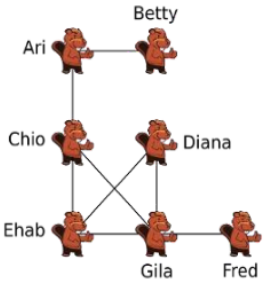
Kezdj egy új játékot, és nyomd le egymás után a D, C, B, B gombokat!

Hol helyezkedik el ezt követően a 4-es szám?

NÉPSZERŰSÉG (2015-CA-01)

Hét hód regisztrált egy online hálózatra. A kép azt mutatja, a hálózaton belül mely hódok „barátok”: a barátok egy vonallal vannak összekötve. A nyári szünet után minden hód megosztott egy fotót a barátaival a hálózaton. Így a barátaik oldalán is megjelent a fénykép. Minden hód a saját oldalán és a barátai oldalán lévő fotókat látja.

Kinek a fényképét látja a legtöbb hód?



VIRÁGOK ÉS NAPOK (2016-CH-12)

Barbara két pecsétet kapott. Az egyik virágot, a másik napot nyomtat. Elgondolta, hogy tudná a nevét csak a virágokkal és napokkal lepecsételni. A különböző betűket virágok és napok különböző sorozatával határozta meg.

Betű	B	A	R	E	Y
Sorozat					

A saját nevét, „Barbara” tehát így pecsételte le:

Ezután lepecsételte angol barátja nevét is:

Hogy hívják a barátját?

BULIVENDÉGEK (2016-IS-02)

Sára bulit szervez. Szeretné meghívni a barátait: Alízt, Bernátot, Cilit, Dávidot és Emilt. De már jól ismeri őket és tudja, néhányan mindig tudni szeretnék, hogy bizonyos barátai eljönnek-e:

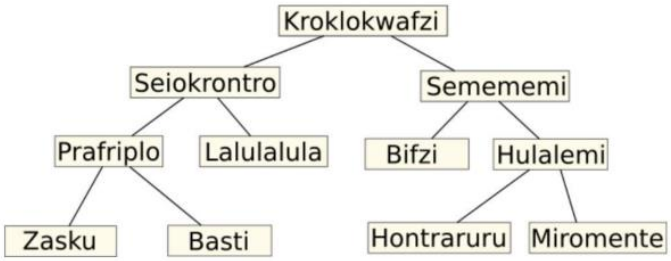
- Dávid tudni szeretné, hogy Alíz jön-e.
- Bernát tudni szeretné, hogy Emil jön-e.
- Cili tudni szeretné, hogy Bernát és Dávid jön-e.
- Alíz tudni szeretné, hogy Bernát és Emil jön-e.

Tehát mielőtt Dáviddal beszélne Alízt kell megkérdeznie.

Milyen sorrendben kérdezze meg barátait Sára?

HIERARCHIA (2016-CZ-03)

Az ábra a Morgenstern bolygó állatainak viszonyát írja le. Két állatcsoport közötti vonal azt jelenti, hogy az alsó típus minden állata egyben a felső állatcsoportba is beletartozik.



Például minden „Hulalemi“ egyben „Sememem“ is. A „Seiokrontro“ egyedei között viszont vannak olyanok, amelyek nem tartoznak a „Basti“ családba. *A következő állítások közül csak egy igaz. Melyik?*

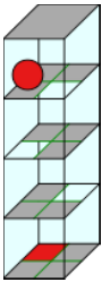
A) Minden Basti egyben Seiokrontro is.

B) Némelyik Hontraruru nem Sememem.

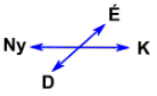
C) Minden Zasku egyben Bifzi is.

D) Minden Prafriplo egyben Basti is.

3D LABIRINTUS (2016-JP-03)



Egy 3D labirintus négy szintből áll, szintenként négy mezővel. A legfelső szinten van egy golyó, amit a legalsó szinten lévő célba (piros mező) kell eljuttatni.



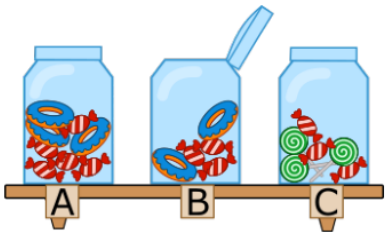
A golyót az É, D, K és Ny utasításokkal irányíthatod (lásd kép).

A fehér mezőkön a golyó egy szinttel lejjebb esik. A labirintus zárt, oldalra nem gurulhat ki a golyó.

Milyen utasításokkal tudod célba juttatni a golyót?

BONBONIER (2016-HU-06)

Károlynak és Juditnak van 3-3 üvegedénye, melyekben édességeket tartanak.

Károly üvegedényei	Judit üvegedényei
	

Minden üvegedény a következő tíz tulajdonságok közül többel is rendelkezhet:

- Vagy nyitva (1) vagy zárva (2) van.
- Vagy tartalmaz piros-fehér csíkos cukrot (3) vagy nem (4).
- Vagy tartalmaz kék cukorfánkot (5) vagy nem (6).
- Vagy tartalmaz zöld spirál-nyalókát (7) vagy nem (8).
- Vagy gömbölyű (9) vagy szögletes (10).

Károly „A” üvegedényének például a 2, 3, 5, 8 és 10-es tulajdonságai vannak.

Nézd meg jól! Károly üvegedényeinek vannak közös tulajdonságaik. Judit üvegedényeinek is vannak közös tulajdonságaik.

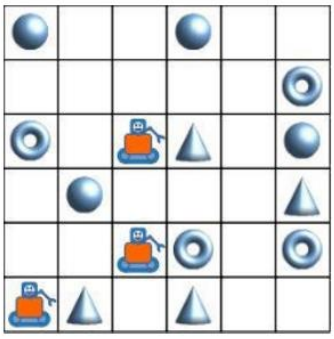
Egy üvegedénynek azonban Károly üvegedényeinek közös tulajdonságai és Judit üvegedényeinek közös tulajdonságai is megvannak. Melyik ez az edény?

EGYSZERRE (2016-IE-05)

Három robot csapatban dolgozik együtt. A csapatot a következő parancsokkal irányíthatod: **N, S, O** vagy **W**. Egy paranccsal mindhárom robotot egyszerre irányítod: egy mezővel tovább a megadott irányba.

Amikor a robotok egy tárgyhoz érnek, azt felveszik. Hogy ne vegyenek fel felesleges tárgyakat, úgy kell irányítanod őket, hogy azokat kikerüljék.

Például: ha a robotokat a **N, N, S, S, O** parancsokkal utasítod, akkor összesen két kúpot és egy karikát vesznek fel.



Ha azt szeretnénk, hogy pontosan egy labdát, egy karikát és egy kúpot vegyenek fel a robotok, melyik parancssorral utasítsuk őket?

A) N, O, O, O

B) N, O, O, S, O

C) N, N, S, O, N

D) N, O, O, S, W

EBÉDSZÜNET (2016-LT-03)

Ebédszünete alatt (12:00-13:00) Alexandra a következőket szeretné elintézni:

- venni egy könyvet a könyvkereskedésben;
- venni egy üveg tejet az élelmiszerüzletben;
- az újonnan vett könyvet feladni a postán;
- inni egy kávét a kávézóban.

E négy tevékenység mindegyikéről Alexandra meghatározta, mennyi időt vesznek igénybe. Az alábbiakban listázott időtartamok azonban csak a csúcsidőn kívül érvényesek. Ezért Alexandra megpróbálja elkerülni a csúcsidőt.

Hely	Időtartam	Csúcsidő
könyvesbolt	15 perc	12:40 – 13:00
élelmiszerüzlet	10 perc	12:00 – 12:40
posta	15 perc	12:00 – 12:30
kávézó	20 perc	12:30 – 12:50

Milyen sorrendben intézze el Alexandra a teendőit, hogy elkerülje a csúcsidőt?

KIX KÓD (2016-NL-04)

Hollandiában az irányítószámok négyjegyűek: számokat és betűket is tartalmazhatnak. Egy egyedi vonalkódja is van minden irányítószámnak, amit Kix kódnak neveznek.

A Kix kód minden sorában van egy felső rész, két hosszú és két rövid vonással, valamint egy alsó rész ugyancsak két hosszú és két rövid vonással. Középen fedik egymást a rövid vonások. A képen (a táblázatban) láthatod a 0, a 7, a G és az Y Kix kódját.

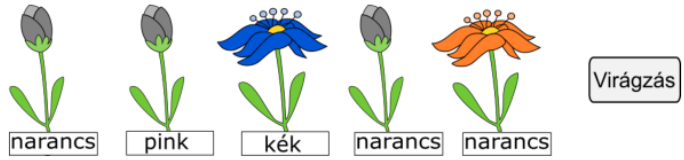
	0	1	2	3	4	5
	6	7	8	9	A	B
	C	D	E	F	G	H
	I	J	K	L	M	N
	O	P	Q	R	S	T
	U	V	W	X	Y	Z

A G7Y0 irányítószám Kix kódja tehát így néz ki:

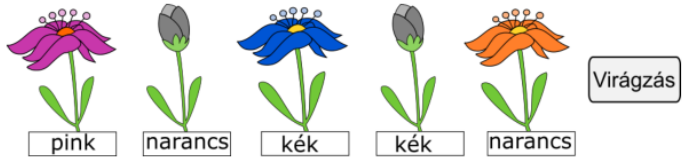
Milyen irányítószámot jelöl az alábbi Kix kód?

VIRÁGZÁS (2016-SK-04)

Jana egy számítógépes játékkal játszik A számítógép öt virágnak titokban választott színt. A lehetséges színek a kék, a narancssárga és a pink. A választott színek nem változnak a játék során. Jana is mindegyik virághoz kiválasztott egy színt és a Virágzás gombra kattintott. A helyes színnel kijelölt virágok kinyíltak, míg a többi nem.



Ezután Jana néhány virág színét megváltoztatta és újra a Virágzás gombra kattintott. A második próbálkozás végeredményét az alábbi kép mutatja.



Melyik elrendezésben választhatta ki a számítógép a virágok színeit?