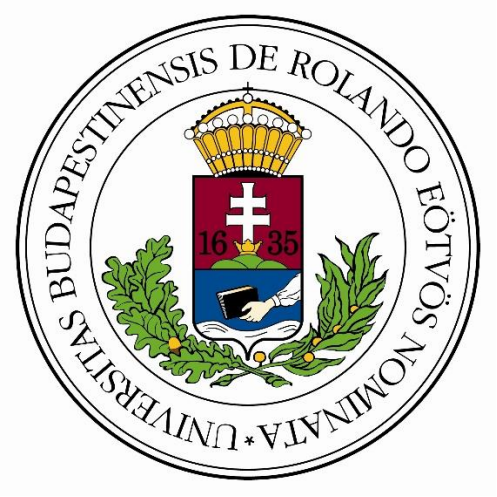


TÖRLEY GÁBOR

VIZUALIZÁCIÓ A PROGRAMOZÁSTANÍTÁSBAN

DOKTORI ÉRTEKEZÉS TÉZISEI

Eötvös Loránd Tudományegyetem Informatika Doktori Iskola

Az informatika alapjai és módszertana Doktori program

Iskolavezető: Dr. Benczúr András
Programvezető: Dr. Demetrovics János
Témavezető: Dr. Szlávi Péter

Budapest, 2013.

Bevezetés

A kutatásom oktatás-módszertani jellegű, és az algoritmikus gondolkodás, a probléma-megoldó képesség fejlesztésére fókuszál.

A kutatás kiindulópontjai a következők:

1. A PISA-jelentések[1] és más hazai mérések[2] szerint a mai magyar tizenévesek a problémamegoldó-képesség, a kreativitás tekintetében csak a középmezőnybe tartoznak a nemzetközi összehasonlításokban. Ezért aktuális oktatási cél a tanulók problémamegoldó gondolkodásának mielőbbi és minél hatékonyabb fejlesztése.
2. A jelenlegi középiskolai tantervek a programozásra legfeljebb 20%-nyi óraszámot szánnak – az egyébként is indokolhatatlanul kevés – informatika-órákon belül. Néhány gimnáziumban van programozásoktatás, de jellemzőbben a szakközépiskolákban tanítanak programozni.
3. A programozáshoz használt nyelvek is – többnyire – valamely, DOS-os környezetben futó magasszintű programozási nyelvet jelent, pl. Pascal, C, C++, BASIC.
4. A programozás köztudottan a legnehezebb fejezete az informatikának, különösen, ha az illető tanuló nem is motivált ennek elsajátításában.
5. A programozásoktatás egyik alkalmas eszközének látszik az 1.-ben megfogalmazott probléma megoldására.
6. A magyar közoktatás céljai között egyre nagyobb hangsúlyt kap a tanulók kognitív képességének fejlesztése, értelmének kiművelése. Ahhoz, hogy bárki elboldoguljon az „Élet útvesztőjében", szüksége van megfontolt gondolkodásra, hogy a napi problémákra gyors és hatékony megoldást találjon. Az *algoritmikus gondolkodás* elsajátítása ebben is nyújt segítséget.

Az algoritmus tervezése folyamán a tanuló sorrendbe fűzi a konkretizált gondolatokat (*algoritmus*), majd elemzi, osztályozza ezeket, mérlegeli saját stratégiáját, végül értékeli megoldását, hogy teljességben lássa a megoldandó problémát, és akár rossz irányú próbálkozásokon keresztül eljusson a megoldásig. Felidézi tapasztalatait, elraktározza, majd felhasználja azokat a következő tervezésnél. Az algoritmusok tudatos alkotása tehát fejleszti a tanulók kognitív képességeit.

[1] Executive Summary PISA 2006: *Science Competencies for Tomorrow's World* © OECD 2007
[2] Molnár Gyöngyvér: *A komplex problémamegoldó képesség fejlettségét jelző tényezők*, In. Magyar Pedagógia, 103. évf., 1. szám pp. 81-103., Szeged, 2003.

A programozásoktatás a programozási tételek tanításával szerepet tud vállalni a tanulók kognitív képességeinek fejlesztésében, mégis, hazai és külföldi tapasztalatok szerint[3] [4], nehéz tanulni és tanítani ezeket. A tanítási módszerek fejlesztése ezen a területen, tehát fontos feladata a mai informatikaoktatásnak. [5] [6]

A témaválasztás indoklása

2005-ben, a diplomamunkám keretein belül végeztem egy kérdőíves vizsgálatot 70 tanuló részvételével. Ennek egyik eredménye: a megkérdezett diákok alig harmada gondolta azt, hogy hasznos számára a tankönyv a programozás tanulásában. Ez felveti azt a kérdést, hogy miért van ez így, továbbá, hogy egy új eszköz bevetésével eredményesebbé lehet-e tenni az oktatást, ki lehet-e egészíteni a tankönyv szöveges, illetve a tanár szóbeli információit?

Tanári tapasztalataim szerint a legnehezebb része a programozási tételek tanulásának, mikor arra a kérdésre keressük a választ, hogy miért is lesz jó az adott algoritmus, miért fogja az adott tétel „megoldani" a problémát. Tehát az adott algoritmus helyességének „bizonyítása" – ezen a szinten természetesen nem matematikai eszközökkel – nehézségekbe ütközik, mert ehhez a tanulónak át kell látni az algoritmus lépéseit, és a lépések közötti összefüggéseket. További probléma (sőt probléma-kettős) az, hogy ha már megértette a tanuló a tétel algoritmusát, akkor abból hogyan is „keletkezik" a konkrét feladathoz tartozó konkrét algoritmus, majd azon is továbblépve a helyesen működő kód. Azaz egyetlen problémába sűrítve mindezt: az *absztrakt* tételt hogyan fogja az adott feladatra alkalmazni, *konkretizálni*.

Tanítási gyakorlatom alatt programozási tételeket tanítottam egy fővárosi szakközépiskolában. Akkor egy prezentációba ágyazott animáción keresztül mutattam meg a diákoknak az adott tétel működését, tehát a diákok látták az animációt, olvasták az algoritmus szövegét, és hallották a magyarázatot. Sokkal többet tudtam így átadni, mint ha csak az algoritmus szövegét tanulmányoztuk volna végig, mert több csatornán, több dimenzióban tudtam

[3] Matti Lattu, Veijo Meisalo, Jorma Tarhio, *A visualisation tool as a demonstration aid*, Computers & Education, v.41 n.2, p.133-148, September 2003.
[4] Urquiza-Fuentes, J., and Velázquez-Iturbide, J. (2013): *Toward the effective use of educational program animations: The roles of student's engagement and topic complexity*, Computers & Education, vol. 67, pp. 178 - 192
[5] Szlávi Péter: *A programkészítési didaktika kérdései.* (Doktori disszertáció) ELTE, 2004.
[6] Szlávi Péter: *Programkészítés és gondolkodás.* Informatika a felsőoktatásban 2008 Konferencia, Debrecen

„közvetíteni” a tananyagot. A *vizuális* és *hallható* csatornán túl az *idő* dimenzióra is építettem, ugyanis az algoritmus időbeli változását is figyelemmel kísérhették a hallgatók. Ezt a hatást hívja Mayer „Multimédia hatásnak”[7].

Tapasztalataimból azt a következtetést vontam le, hogy ha olyan eszközöket használok a tanítás alatt, amelyek segítenek *elképzelni* az algoritmus belsejében történteket, akkor ezek meggyorsítják a megértés folyamatát. Az algoritmus-vizualizációs (AV) eszközök használata ebben nyújthat segítséget.

A programozástanítás egyik szegmense, az objektum orientált programozás (OOP) az elmúlt évtizedekben a leghatásosabb programozási paradigmává vált. Széles körben használják az oktatásban, az iparban, és majdnem minden egyetem tantervében előfordul az objektum orientáltság fogalma. A programozói közösségek szerint is hasznos OOP-t tanítani, mert ez a paradigma támogatja a strukturált programozás elveinek rögzülését, a modularizáció és a programtervezés koncepciójának tanítását.

A fent említett strukturáltság mellett az OOP-s szemlélet természetes módon képes a valós világot leképezni, hiszen a körülöttünk levő „dolgok” is leírhatók tulajdonságokkal, állapotokkal és állapotváltozásokkal (metódusokkal), ilyen módon az objektum orientált paradigma fejleszti az absztrahálás képességét.

Azonban az OOP tanítása, tanulása komolyabb kihívás jelent a tanárnak, a diáknak egyaránt, egyrészt az absztrakt gondolkodás szükségessége miatt (osztály fogalom), illetve azért is, mert az objektum, amit a programozás során létrehozunk, csak a programozási folyamat végére konkretizálódik, ölt testet.

Egy objektum működésének megértése, majd kódolása, a fenti tapasztalat miatt kevesebb időbe kerülhet, ha vannak olyan eszközeink, amelyek segítenek abban, hogy elképzeljük az adott objektumot.

Kölling szerint[8] néhány hallgatónak azért nehéz megérteni az OOP terminológiáját, mert mindaz, amit a képernyőn és a tanár prezentációjában lát, az maga a programkód. Ha elvárás a hallgatótól, hogy gondolkodni és beszélni tudjon az osztályokról, a kapcsolataikról, akkor

[7] Mayer, R. E. (1997). *Multimedia learning: Are we asking the right questions.* Educational Psychologist, 32, 1-19.

[8] Kölling, M., "The Problem of Teaching Object-Oriented Programming, Part 2: Environments," *Journal of Object-Oriented Programming*, 11(9): 6-12, 1999.

vizuálisan célszerű azokat megjeleníteni. Ugyanez az állítás igaz, amikor a program futási idejében vizsgáljuk az objektumokat. Vizuális reprezentációval könnyebben meg lehet érteni az OOP terminológiáját.

Előzmények és célkitűzések

A 2007/2008-as tanév őszi szemeszterét Helsinkiben töltöttem, az Erasmus ösztöndíjprogram keretein belül. Kutatást végeztem a finnországi informatika- és (azon belül) programozásoktatási módszerekről. Órát látogattam, középiskolában, illetve egyetemen tanító tanárokkal készítettem interjúkat. Magam is részt vettem programozásoktatásban, mint hallgató. Itt találkoztam egy új eszközzel, a Jeliot-tal, és egy rajta nyugvó oktatási elképzeléssel, az algoritmus vizualizációval. Ezzel az eszközzel és ezen a módon támogatják a programozás tanítását és tanulását középiskolai szinten Izraelben és az Amerikai Egyesült Államokban. Tapasztalataimról jelentést írtam.[9]

A kutatásom célja, hogy részletesen bemutassak egy algoritmus vizualizációs (AV) eszközt, és egy ehhez kapcsolódó vizsgálati és tanítási módszert, amely különösen az átlagos képességű hallgatókat segíti a felzárkózásban.

Az alkalmazott módszerek

A 2012/2013-as tanév őszi félévében egy kétcsoportos pedagógiai vizsgálatot folytattam az ELTE Informatikai Karán, ahol a Programozási alapismeretek kurzusban résztvevő programtervező informatikus hallgatók egész féléves munkáját vizsgáltam. Az évfolyam hallgatói közül 5 csoport vett részt a kísérleti csoportban és 5 csoport vett részt a kontroll csoportban, összesen 153 fővel. A létszámokat az alábbi táblázat mutatja:

	Kísérleti csoportok	Kontroll csoportok
	16	18
	15	16
	16	14
	16	15
	15	12
Összesen:	78	75

[9] Gábor Törley: ICT in Finland - http://pezsgo.web.elte.hu/publikaciok/ict_in_finland.pdf - Letöltve: 2013. augusztus 7.

A kutatás első mozzanata egy előzetes tanulmányokat felmérő kérdőív kitöltetése volt a kísérleti és a kontroll csoport hallgatóival (a kérdőív a disszertáció Függelékében található). A kérdőív célja volt egyrészről számszerűsíteni azt, hogy ki, mennyi órában tanult informatikát, illetve programozást középiskolai tanulmányai alatt, másrészt felmérni a hallgatók algoritmikus gondolkodásának állapotát még azelőtt, hogy a tantárgy hatással lenne arra.

A kutatás folyamán az algoritmikus gondolkodásról szóló 3-5. kérdésre a hallgatók pontszámot és jegyet kaptak, ez alapján lehet mérni a hallgatók fejlődését. Az 5. kérdés a hallgató absztrakciós készségét mérte. A kérdések kiértékelésénél az azokra adott válasz helyességét és részletességét vizsgáltam. A részletesség azért fontos a helyesség mellett, mert ezáltal lehet mérni a gondolkodásmód sokrétűségét: mennyire figyel a hallgató az apróbb részletekre. Annál több pont járt a feladat részletességére, minél fontosabb előfeltételt adott meg a feladat megoldója. A három feladatra maximum 14 pontot lehetett kapni.

Ahhoz, hogy a feladatok eredményéhez képest meg lehessen határozni a fejlődést, ötfokozatúvá kellett alakítani az értékelést, hasonlóan a későbbi számonkérések eredményeihez. Pontozási rendszer:

alsó pont	**jegy**	**arány**
0	1	0%
5,5	2	39%
7,5	3	54%
10	4	71%
12	5	86%
14	(max)	

A bemeneti feladat jegyének meghatározása után már van kiindulópontunk ahhoz, hogy meghatározzuk a fejlődés mértékét. A félév folyamán a hallgatók négy számonkérésen adnak számot a tudásukból:

- 1. csoportzárthelyi: két egyszerű programozási tétellel megoldható feladat kódolása, specifikáció és algoritmus alapján (a számonkérés célja: dominánsan algoritmusértés + kódolás ismeret),
- 2. csoportzárthelyi: három egyszerű programozási tétellel megoldható feladat algoritmizálása (struktogrammal), specifikáció alapján (a számonkérés célja: specifikációértés + algoritmuskészítés, ezek együtt jóval több absztrahálási képességet kívánnak, mint az előbbi),

- beadandó feladat készítése, ahol négy részfeladatot kellett megoldani, amelyek közül az egyikben előfordultak összetett programozási tétellel megoldható feladatok (a számonkérés célja: komolyabb méretű feladat igényes megoldása, „termék-gyártás”),
- Évfolyam zárthelyi dolgozat, amelynél a négy részfeladatból 2 volt olyan, amelyet összetett programozási tétellel kellett megoldani (komolyabb méretű probléma teljes megoldása).

A fejlődés meghatározásánál nem vettem figyelembe a beadandók eredményét, ugyanis ennél a számonkérésnél kevésbé garantálható, hogy a hallgató önálló munkát adott be, ilyen módon a hitelesség megkérdőjelezhető. A többi három számonkérést az időbeli sorrend alapján súlyoztam az értéküket, ugyanis minél jobban távolodunk az első gyakorlat idejétől, annál mélyebbnek kellene lennie a megértésnek. A fejlődés mértékét (F) a következő képlettel határoztam meg:

$$F = [(CsZH_1 - E) + 1{,}5 * (CsZH_2 - E) + 2 * (\acute{E}vfZH - E)]/18,$$

ahol $CsZH_1$ az 1. csoportzárthelyi jegye, $CsZH_2$ a 2. csoportzárthelyi jegye, $\acute{E}vfZH$ az évfolyam zárthelyi jegye, illetve E az előzetes tanulmányok 3-5. kérdésére kapott osztályzat. A fenti képlet eredménye egy -1 és 1 közötti valós szám lesz. Az eredmény előjele mutatja a fejlődés irányát, az értéke pedig a nagyságát.

Az algoritmikus gondolkodást felmérő 3-5. kérdések eredményei hasonlóan alakultak mindkét csoportnál, nem tapasztalható jelentős eltérés, tehát kimondható, hogy a két csoport közel azonos szintről indul, tehát a későbbiekben közlendő fejlődést meghatározó értékek közötti különbség nem a bemeneti többlettudásnak köszönhető.

A dolgozat tézisei

Hipotézis 1.: Az AV beilleszthető a közoktatási informatikába

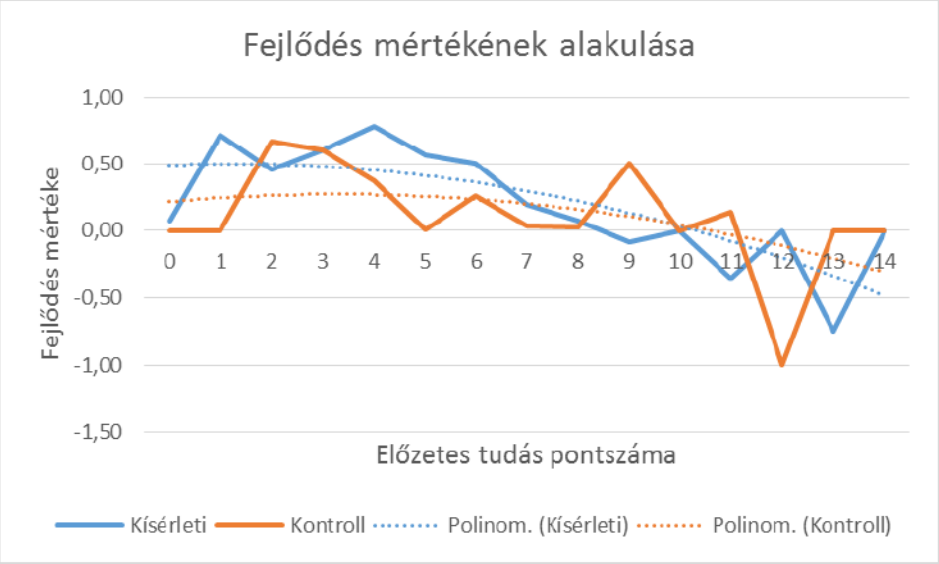
A közoktatásban kötelezően alkalmazandó informatika kerettanterv a „Problémamegoldás informatikai eszközökkel és módszerekkel” c. tematikai egységének célja az algoritmizálási készségek fejlesztése, mert a valós élet tevékenységei, kezdve a bevásárló listától az utazásig, döntések sorozatából épül fel.

A dolgozatomban bemutatok alkalmazási példákat, illetve a kerettanterv azon téma-köreire, amelyeknél alkalmazni lehet az AV-t, egy lehetséges órafelosztást, az Nemzeti Fejlesztési Minisztérium által ajánlott óraszámokat alapul véve.

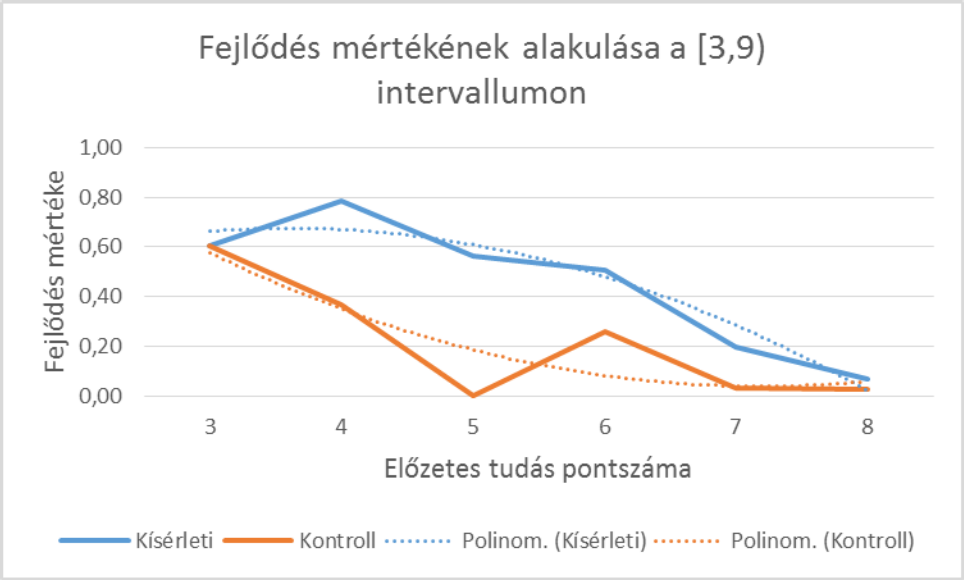
A cél az órai problémák animációval való kiegészítése, hogy a tanulók fel tudják azt használni algoritmusok megértésénél, készítésénél, majd később a kódolásnál is, illetve, hogy a tanár az oktatása közben fel tudja használni az animáció adta lehetőségeket, különösképpen az animáció folyamatosságát.

Hipotézis 2.: Az AV eszközzel való tanítás az átlag közeli tudású hallgatók felzárkózását segíti

A hallgatók túlnyomó része, 84%-a a [3,9) intervallumban szerzett pontot az előzetes tudást felmérő feladatokon. Az eredmények alapján a kísérleti csoport 78%-a, míg a kontroll csoport 89%-a esik bele ebbe az intervallumba. A fejlődés mértékét az alábbi két diagram mutatja, a második a kiemelt [3,9) intervallumét:



1. ábra. *Fejlődés mértékének alakulása.*

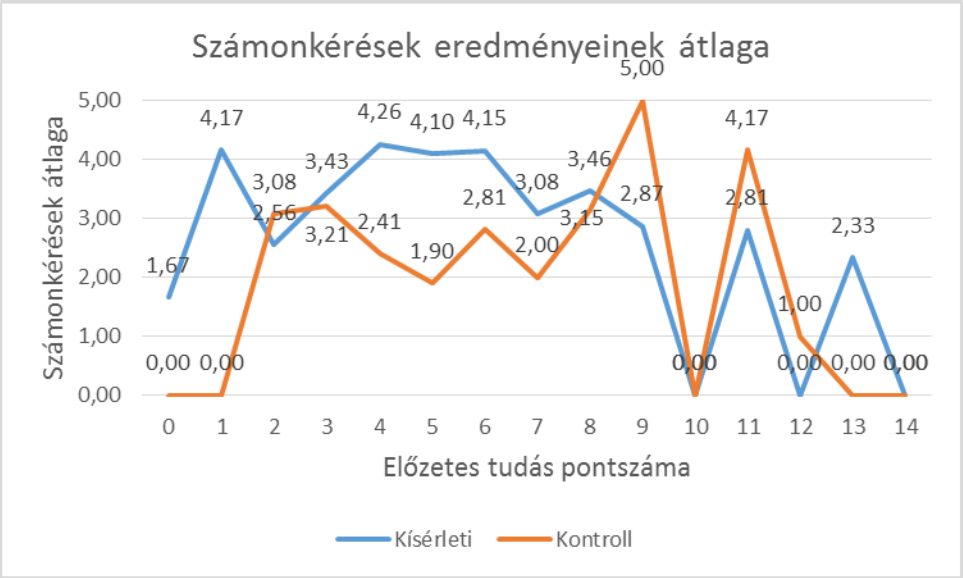


2. ábra. *Fejlődés mértékének alakulása a [3,9) intervallumon.*

A fenti diagramok alapján levonható az a következtetés, hogy a kísérletben résztvevő hallgatók 84%-ánál az AV-t használó csoportok nagyobb fejlődést értek el, mint a kontroll csoport hallgatói. Különösen az átlagos, illetve az alatti hallgatók felzárkózását segítette igazán az AV rendszerrel való tanítás. A kísérlet alapján azokat a hallgatókat nevezzük átlagos képességűnek, akiknek az előzetes tudást felmérő feladatokra kapott pontszámuk az [5,6) intervallumban volt. Ennek oka, hogy a kísérletben résztvevők átlagos pontszáma 5,74 volt. A legmagasabb fejlődést (0,79) azonban a [4,5) intervallumban pontszámot kapó hallgatók érték el a kísérleti csoportban.

Hipotézis 3.: Az AV-t használó hallgatók többet fejlődnek és jobb eredményeket érhetnek el, mint a vizualizációt nem használók

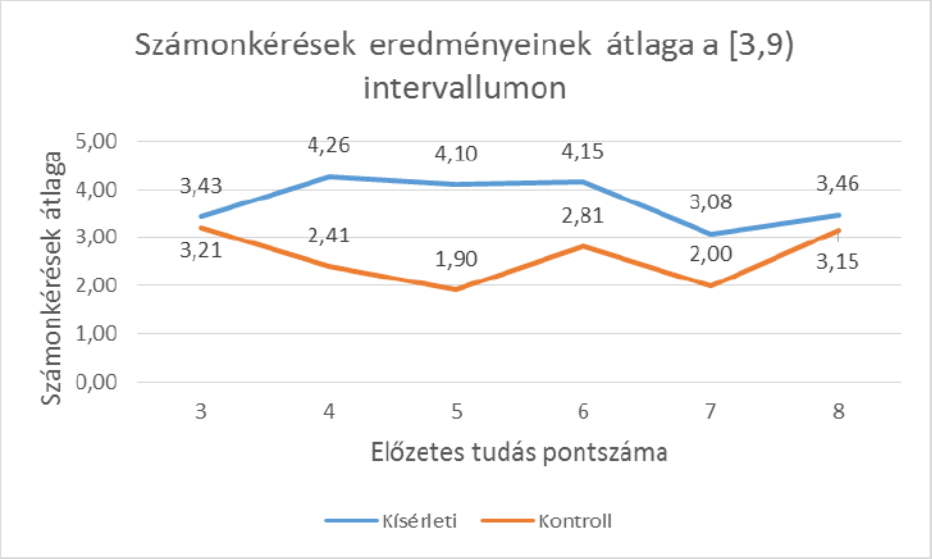
Külön vizsgálat tárgyát képezte a félév végi eredmények összehasonlítása.



3. ábra. *Számonkérések eredményeinek átlaga.*

Az eredmények vizsgálatánál nem a gyakorlati jegyeket vettem figyelembe, ugyanis a számonkérések átlaga jobban tükrözi a hallgató teljesítményét.

Látható, főként a hallgatók jelentős többségét magába foglaló [3,9) intervallumban, hogy különösen az átlagos hallgatóknál volt szignifikáns a különbség a kísérleti csoport javára. Ez az eredmény összefüggésben van a fejlődéssel is, tehát az átlagos előzetes tudású hallgatók felzárkózására és eredményeire volt pozitív hatással az AV eszközzel való tanítás.



4. ábra. *Számonkérések eredményeinek átlaga a [3,9) intervallumon.*

A lenti táblázatból látható, hogy az AV programot használók értek el jobb átlagot, majdnem egy jeggyel jobbat, mint a kontroll csoport hallgatói.

	Kísérleti csoport	Kontroll csoport
Számonkérések átlaga	3,58	2,65

Az AV programmal való tanulás hatással volt bukási arányra, ugyanis szignifikánsan kisebb arányú hallgató bukott meg a kísérleti csoportban, mint a kontroll csoportban. A kísérleti csoportban a hallgatók kétharmada legalább 3,5-ös átlagot ért el a számonkéréseken (a kontroll csoportban csak 37%).

Hipotézis 4.: Az AV használata jobban fejleszti az absztrakciós készséget, mint ha nem kerülne használatra ez az eszköz

Az AV absztrakt készségre gyakorolt hatását az 5. kérdés és a 2. csoportzárthelyi dolgozat eredményei mutatják meg. Az 5. kérdésre kapott pontszám mutatja meg, hogy a félév első hetében hol állt ez a készség, illetve a 2. csoportzárthelyi pedig azt, hogy fejlődött-e.

	Kísérleti csoport	Kontroll csoport
5. kérdés átlagos pontszáma	1,60	1,67
2. csoportzárthelyi átlaga	4,13	3,18

A fenti táblázat szerint a kísérleti csoport valamivel hátrébbról indult, mint a kontroll csoport, mégis majdnem 1 jeggyel jobb átlagot produkált, mint a kontroll csoport, ami azt jelenti, hogy az AV használata jobban fejleszti az absztrakciós készséget, mint ha nem került volna használatra ez az eszköz.

A szerző disszertációhoz kapcsolódó publikációi

- Törley, Gábor: *Expressiveness of programming languages and environments: a comparative study*, in. Teaching Mathematics and Computer Science Vol. 6/Infodidact, pp. 111-141., 2008., Institute of Mathematics, and Faculty of Informatics University of Debrecen, Hungary
- Törley, Gábor: *Algorithm visualization in programming education*, in. Journal of Applied Multimedia Vol. 4, No. 3., pp. 68-80., 2009., Apple Hungarian VAD, Budapest, Hungary

- Szlávi, Péter – Törley, Gábor: *Teaching Sorting in ICT*, in. Teaching Mathematics and Computer Science Vol. 7, No. 1., pp. 101-117., 2009., Institute of Mathematics, and Faculty of Informatics University of Debrecen, Hungary
- Törley Gábor: *Problémaalapú tanítási módszer a programozásoktatásban* (Info Savaria Konferencia 2006., Szombathely)
- Szlávi Péter - Törley Gábor: *Rendezések oktatási lehetőségei az informatika tantárgyban* (Info Savaria Konferencia 2007., Szombathely)
- Törley Gábor: *Algoritmus-vizualizáció a programozás-oktatásban* (Info Savaria Konferencia 2008., Szombathely)
- Törley Gábor: *Programozási nyelvek, környezetek kifejezőerejének vizsgálata* (Infodidact Informatika Szakmódszertani Konferencia 2008., Szombathely)
- Törley Gábor: *Turbo Delphi használata az oktatásban* (Info Savaria Konferencia 2009., Szombathely)
- Törley Gábor: *Algoritmus-vizualizáció a programozás-oktatásban* (Multimédia az oktatásban Konferencia 2009., Debrecen)
- Törley Gábor: *Objektum orientált programozás Jeliot-tal* (Info Éra Konferencia 2010., Füzesgyarmat)
- Törley Gábor: *Objektum orientált programozás tanítása vizualizációs eszközökkel* (InfoDidact Konferencia 2012., Zamárdi)