Óbudai Egyetem
Kandó Kálmán Villamosmérnöki Kar
Trefort Ágoston Mérnökpedagógiai Központ

DIPLOMAMUNKA

OE-KVK-TMPK
2020

Hallgató neve: **Gombos Szabolcs**
Hallgató törzskönyvi száma: **T009202/FI12904/K**

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Óbudai Egyetem
Kandó Kálmán Villamosmérnöki Kar
Trefort Ágoston Mérnökpedagógiai Központ

DIPLOMAMUNKA FELADATLAP

Hallgató neve: Gombos Szabolcs
Diplomamunka száma: SZD22032414207461
Törzskönyvi száma: T009202/FI12904/K **Neptun kódja:** [redacted]

Szak: tanári [4 félév [mérnöktanár]]
Specializáció:

A dolgozat címe: A problémamegoldó képesség fejlesztésének szerepe a programozásoktatásban
A dolgozat címe angolul: The role of improving problem solving skill in the programming education
A feladat részletezése:

1. Problémamegoldás az oktatásban
2. A programozás és a matematika közötti kapcsolat
3. Informatika érettségi feladatok megoldása halmazokkal
4. Esettanulmány az analógiák szerepéről a nyelvi elemek elsajátításában

Intézményi konzulens neve: Dr. Holik Ildikó Katalin

A kiadott téma elévülési határideje: 2023. 05. 15.
Beadási határidő: 2022. 05. 15.

A diplomamunka: Nem titkos.

Kiadva: Budapest, 2022. 03. 29.

Trefort Ágoston Mérnökpedagógiai Központ ÓBUDAI EGYETEM

P.H.

.....
Intézetigazgató

A dolgozatot beadásra alkalmasnak találom:

.....
belső konzulens

.....
külső konzulens

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

KANDÓ KÁLMÁN VILLAMOSMÉRNÖKI KAR

TREFORT ÁGOSTON MÉRNÖKPEDAGÓGIAI KÖZPONT

HALLGATÓI NYILATKOZAT

Alulírott hallgató kijelentem, hogy a diplomamunka saját munkám eredménye, a felhasznált szakirodalmat és eszközöket azonosíthatóan közöltem. Az elkészült diplomamunkában található eredményeket az egyetem és a feladatot kiíró intézmény saját céljára térítés nélkül felhasználhatja, a titkosításra vonatkozó esetleges megkötések mellett.

Budapest, 2022. május

Gonda Szabolcs

.....

hallgató aláírása

MEMBER OF eua EUROPEAN UNIVERSITY ASSOCIATION

1081 Budapest
Népszínház u. 8.

+36 1 666-5389

simonics.istvan@tmpk.uni-obuda.hu
www.uni-obuda.hu

Tartalomjegyzék

1. Bevezetés

Gyerekkorom óta érdekel a programozás világa. A tanulmányaim megválasztása során tudatosan a szoftverfejlesztő szakmára készültem fel. Először a szakközépiskolában találkoztam a programozás tantárggyal, majd az BSc jelentkezés során is olyan szakot választottam, ahol elmélyíthettem a tudásomat a programnyelvek és az algoritmusok témakörében. Az alapfokú képzés elvégzése után szoftverfejlesztőként helyezkedtem el, és az ipar különböző szegmenseiben tettem próbára a képességeimet.

Az iparban eltöltött évek során egyre gyakrabban merült fel bennem az igény az elsajátított tudásomnak az átadására. Tapasztaltabb szoftverfejlesztőként előbb gyakornokok és pályakezdők tanítását vállaltam el, majd később a magántanítványaimmal való foglalkozás során gyakoroltam a mentorálást, valamint a tanári szerepkört.

Az oktatási tevékenységeim során megpróbáltam felismerni a tudás átadásának cselekményében azokat a struktúrákat, amelyek segítségével eredményesebben tudom a tananyagaimat megtervezni. A laikus hozzáállásom oka elsősorban az iparban szerzett gyakorlatom, mert programozás közben is a generikus megoldások megértése és alkalmazása nyújtott előrelépési lehetőségeket a karrieremben. Elsőként azt figyeltem meg, hogy hiába mutatom meg példákon keresztül programozási mintákat, ha azt a tanítványaim a későbbiekben nem tudják majd alkalmazni. Ekkor szembesültem a tanításmódszertani és didaktikai ismereteimnek a hiányosságaival. Elhatároztam, hogy bővítem a tudásomat az említett területeken egy mérnöktanári képzésen. A taxonómia rendszerek (Bárdossy 2006: 117) megismerését követően körvonalazódni kezdett számomra egy struktúra, amelyet programozás gyakorlatának oktatása során lehetséges és érdemes volna alkalmazni.

A neveléstudomány pszichológiai vonatkozású szakirodalmának tanulmányozása közben felkeltette az érdeklődésemet néhány olyan kutató, akik a logika fejlesztésének lehetőségeit és a matematikai ismeretek elsajátításának pszichológiai hátterét vizsgálták. Az általam tanulmányozott publikációk elemzése során arra a következtetésre jutottam, hogy a matematika oktatás kutatásában elért eredmények a programozás oktatásban is hasznosíthatóak lehetnek.

A dolgozatom célja, hogy a programozás műveleteinek bemutatása során a matematika eszköztárának felhasználásával lemodellezzek egy olyan problémamegoldási módszert, ami a szoftverfejlesztésben alkalmazható. Személyes tapasztalataim szerint a programozás oktatásában a problémamegoldás területe – ezáltal a matematikai ismeretek alkalmazása is – háttérbe szorul, és inkább a tanítási gyakorlatban a specifikus tudás átadása van a fókuszban. A programozás oktatás bőséges szakirodalma helyett én elsősorban matematikáról szóló

publikációkat használtam fel a dolgozatomhoz. Ezt abból a megfontolásból teszem, mert elsősorban a tantárgy oktatásának kezdeti szakaszára koncentrálok, amikor a diákoknak a korábban szerzett problémamegoldási tapasztalataikra alapozva kell felépíteniük a szoftverfejlesztéshez szükséges gondolati struktúrákat.

A dolgozatom „Problémamegoldás az oktatásban" fejezetében a programozás oktatási gyakorlatának rövid ismertetését és a curriculumfejlesztéshez kapcsolódó alapfogalmak áttekintését követően kitérek a Nemzeti Alaptanterv (NAT 2020) digitális kultúra és a matematika tantárgy tanításának céljára és alapelveire.

Tanulmányaink során találkozhatunk olyan ismeretekkel, amelyek tudása előfeltétele egy másik új ismeret megszerzésének. A tanulás szempontjából az előbbit eszköz-ismereteknek, míg az utóbbit cél-ismereteknek nevezzük. A cél-ismeretek megtanulhatóak puszta memorizálással az eszköz-ismeretek tudása nélkül is, ám ebben az esetben amikor megváltoznak a feladat által definiált feltételek, akkor egy csakis cél-ismeretekkel rendelkező ember nem tudja alkalmazni a tudását. (Orosz 1986: 43-44)

Az előbbi összefüggések bővebb bemutatására teszek kísérletet a dolgozatom második, „A programozás és a matematika közötti kapcsolat" című fejezetében. A matematika tanulmányok során elsajátítható gondolkodási műveleteket mutatom be, amelyeket a problémamegoldás során alkalmazni tudunk.

A dolgozatomban a szimbólumok jelentésének összefoglalása után bevezetek egy absztrakt platformot, amellyel korábbi érettségi vizsga feladatsorok megoldását szemléltetem. Ezzel az a célom, hogy bemutassam az algoritmizálás lépéseit a programozáshoz tartozó tartományspecifikus fogalmak mellőzésével.

A dolgozatom utolsó hosszabb egységében az esettanulmányaim bemutatásával szemléltetem, hogy a programozásban kezdő tanulócsoportoknál az analógiák segítségével hogyan sikerült megkönnyítenem a programnyelv nyelvi elemeinek elsajátítását.

2. Problémamegoldás az oktatásban

Napjainkban a programozás oktatás rendkívül népszerű. Ennek oka, hogy az ipar igényeit a jelenleg rendelkezésre álló szakemberek csak részben tudják kielégíteni. A munkaerőpiac a megjelenő szoftverfejlesztő hiányra úgynevezett *bootcampek* (Hazai bootcampek 2022) létrehozásával reagált. Ezek a *bootcampek* a jelentkező – többségében felnőtt – tanulót a biztos állás, és a rekord idő alatt elsajátítható programozási készségek ígéretével csábítja magához. Egy ilyen képzés időtartama kezdetben egy-két év volt, de már találkozhatunk alig több mint 6 hetes képzéssel is. Általában a képzések célja kimerül abban, hogy a programozáshoz szükséges tudást adják át: „A képzés végére junior szintű programozói tudást szerzel, valamint sok-sok gyakorlatot különböző technológiákban.”(Greenfox 2022)

A jövő programozóit is hasonló ígéretekkel csábítják a szoftverfejlesztői pálya választására a különböző robotépítő- és programozó táborokon, valamint különböző könyveken keresztül is (Whitney 2021, Woodcock 2017). A táborok segítik a belső motivációk kialakítását, és ha egy-egy kiadványt megvizsgálunk, a nyelvezete és illusztrációi játékosan vezetik be a fiatalabbakat is programozás világába. A könyvek ajánlói szintén meggyőzik olvasóikat, hogy elolvasásuk után programozókká válhatnak.

A programozást tanulni vágyóknál a közoktatás az egyik leggyakoribb tanulási környezet. Ennek terveit a Nemzeti Alaptanterv (NAT 2020), valamint a szakképzési jegyzék ide vonatkozó Képzési és kimeneti követelmények (KKK 2020) és Programtervek (PTT 2020) határozzák meg.

A szoftverfejlesztés egy rendkívül szerteágazó terület. Amikor a programozás oktatásáról beszélünk, akkor muszáj a teljes skálának egy speciális szegmensére szűkítenünk a tananyagot (például egy programnyelv keretei közé). Éppen ezért fontos meghatároznunk, hogy pontosan milyen célt is szeretnénk elérni a tananyag feldolgozásának végére. A korábban említett példák eredményességét a meghatározott céljaik elérésében nem vonom kétségbe. Valószínűsíthető, hogy az oktatási programok szerzői is a következő kérdések megválaszolásával kezdtek bele tantervük megalkotásába, ezért magam is kísérletet teszek erre. Milyen célokat tűzhetünk ki a programozás oktatás számára? Hogyan definiálhatjuk a programozói tudást?

A kérdések megválaszolása érdekében először a tudás meghatározását kell vizsgálnunk. Báthory Zoltán meghatározása a következő: „Tudás az információk, készségek, képességek, mozgások cselekvések, magatartás, attitűdök, érdeklődés, szokások, világkép és a legmagasabb fokon világnézet” (Báthory idézi Bredács 2015: 7). Egy tananyag átadásakor azt érdemes

figyelembe venni, hogy a felsoroltak alapján miket szeretnénk lefedni. Itt arra gondolhatunk, hogy célunk-e egy adott attitűd formálása, vagy csupán a szükséges információk átadása.
Egy programozás tananyagot felépíthetünk úgy, hogy egy adott programnyelvvel kapcsolatosan osztunk meg információkat, és érdekes példákon keresztül szemléltetjük azokat. Ezáltal a tanulók megtanulhatják a programnyelvhez kapcsolódó nyelvi elemeket, és a típus feladatokon keresztül begyakorolhatják azokat. Az értékelésekkor ezeket az információkat számon is kérhetjük, amelyek eredményéből megállapíthatjuk, hogy elértük-e a kitűzött célokat (Bredács 2015: 5).
Az előbbi esetben a cél definiálásakor szem előtt kell tartanunk, hogy a tananyagot lekorlátozzuk arra a specifikus tartományra, amelyet számonkérünk. A programozás esetében ez általában egy erősen korlátolt intervallumot jelöl. Ha egy szűk tartományon belül értelmezünk egy programnyelvet, akkor egy adott probléma megoldásához szükséges eszközkészletre szűkíthetjük a tananyagot. Ezzel lerövidíthetjük az oktatás időtartamát.
A munkaerőpiacon bevett gyakorlat a több körből álló szakmai interjú, amelynek egyik része egy programozási nyelv ismerete, míg a másik az általános problémamegoldó készséget méri fel. Az interjúztató célja a szakmai interjúkon általában a nyelvi ismeretek felmérése mellett a generikus programozási készségek felmérése is. Ez utóbbinak az egyik oka, hogy az esetek többségében egy olyan speciális környezetben folyik a szoftverfejlesztés, amelyet csak a cég keretein belül ismernek.
Az ipar efféle számonkéréseire a tantermi körülmények csak részben készítik fel a diákokat. Tanárként törekedni kell arra, hogy az eszközök megismerése mellett egy olyan tudáshalmazt adjunk át, amelyet a tanuló egy új, ismeretlen környezetben is hasznosítani tudjon. Az oktatás céljaként tehát definiálhatjuk annak a tudásnak az átadását, aminek a segítségével a tanuló a jövőben képes a korábbi tapasztalatait beépíteni egy-egy probléma megoldásába. (Pentelényi 2011: 18-22)
Az előbbi megállapításunk meghatározó tényezője a tantervek fejlődési irányának. A gyakorlati- és elméleti tudásanyag egymáshoz való közelítésével elérhetővé válhat, hogy a gyakorlati tapasztalatok segítségével történik az elméleti tudás bővítése. (Ballér 2001: 70-71)

2.1. Curriculumfejlesztés

Napjainkban a leginkább elfogadott tantervelméleti modell az úgynevezett *curriculumelmélet*. Curriculumfejlesztésen annak az oktatási programnak a tervezését és fejlesztését értjük, amely meghatározza a tanulási folyamat egységeit, és leírja a tanítási folyamathoz tartozó célokat és

értékeléseket. A curriculum az oktatásszervezés különböző szintjein határoz meg egy tantárgyhoz tartozó tervet. A legfelső szinten a *core curriculum* rögzíti azokat a tartalmakat, amelyeket egy képzés résztvevőinek el kell sajátítania. (Bárdossy 2006: 114)

A *local curriculum* az oktatásszervezés egy alacsonyabb szintjén határozza meg a tartalmi egységeit a tananyagnak. Ez a fejlesztési terv az intézmény szintjén definiálja az adott képzéshez tartozó értékelési szempontokat, és az oktatás céljait. Általában az adott képzéshez tartozó programterveket egészíti ki az intézmény vagy a pedagógus a helyi sajátosságok figyelembevételével. (Bárdossy 2006: 115)

A *curriculumelmélet* az úgynevezett Tyler-racionálén, vagy más néven Tyler-modellen alapszik, melyet Ralph Tyler dolgozott ki a múlt század közepén. A struktúra meghatároz egy olyan algoritmust, amelynek bemeneti oldalán a tananyag kiválasztásának forrásai és a szűrői találhatóak, míg eredményként a tanulás tervezett eredményeit kaphatjuk meg. A tananyag kiválasztásának forrásai alatt elsősorban olyan tényezőket értünk, mint a társadalom igényei, illetve az átadott tudás relevanciája. A korábbi tantervi műfajtípusokhoz képest megjelenik egy új tényező ugyancsak a forrás oldalon, hogy a tananyag-fejlesztés során figyelembe kell venni a tanulói szükségleteket is. A szűrők alatt olyan tényezőket értünk, melyek közvetlen hatással lehetnek a programtervre, ilyenek a nevelésfilozófiai és a tanuláspszichológiai álláspontok. (Perjés – Vass 2008)

A Tyler-modell a kitűzött célok megtervezésének metodológiája mellett a tanulás-tanítás hatékonyságának megtervezésére is fókuszál. Ezekbe beletartozik az is, hogy a célok és a tervezett tanulási eredmények mellett a tanulási teljesítmény mérésének és értékelésének összefüggéseit is feltárja. A modellben Tyler kiemelt figyelmet szentel az értékelésnek is, amely kimutathatja, hogy a programterv milyen változásokat eredményezett a diákokban. (Bárdossy 2011)

Az értékelés nagyon fontos a problémamegoldás esetén is, melyet a jegyekkel csak nagyon felületesen tudunk elvégezni. (Pólya 1977: 84) Emiatt ha a problémamegoldás képességének fejlesztését helyezzük a programtervünk fókuszába, akkor a tervnek tartalmaznia kell egy szempontrendszert, ami az értékeléseket megalapozza, és a tanulóknak részletesen beszámol az esetleges hibáiról és részeredményeiről. Az elvárt eredményeket, az objektív ellenőrzéshez szükséges szempontokat, és az értékelés menetét követelményrendszerekbe foglalhatjuk (Bárdossy 2011).

A magyar oktatáspolitikában a NAT tartalmazza azokat a tartalmi elemeket, amelyeket minden közoktatásban részesülő tanulónak el kell sajátítania a képzés végére. Arra vonatkozóan, hogy a programozás tantárgy oktatásával kapcsolatosan milyen tartalmi megkötések vannak, a

szakképzési jegyzék szoftverfejlesztő képzéshez tartozó PPT és KKK dokumentumában is találhatunk információkat. A helyi curriculumot ezek alapján kell összeállítania az intézményeknek, és a tantárgyat oktató pedagógusoknak.

2.2. A Nemzeti Alaptanterv részei és oktatás-nevelési szakaszai

A NAT a tantárgyakat műveltségi területek alapján osztja fel (NAT 2020: II.3.). Minden műveltségi területhez meghatározza a következőket: alapelvek és célok, fő témakörök, és tanulási eredmények. Ezek a Tyler-racionáléval azonosíthatóak az alapján, hogy az alapelvek és a célok összefoglalják a tudás relevanciáját a társadalmi elvárásokkal együtt, figyelembe véve a nevelésfilozófiai- és a tanuláspszichológiai álláspontokat. A fő témakörök azokat a tanulási tapasztalatokat foglalják össze, amiket az iskolának biztosítania kell a tanulók számára. Végül a Tyler rendszerében tárgyalt követelményrendszert a tanulási eredmények szekciója alapján fogalmazza meg a NAT.

A programtervben foglalt kulcskompetenciákhoz tartozó műveltségi területek a nevelés-oktatás szakaszai alapján vannak tagolva. A bontás biztosítja, hogy a tananyagok a tanulók életkori sajátosságaihoz illeszkedjenek. Az alapfokú oktatás első- és második négy osztálya, valamint középfokú nevelés négy évfolyama egy-egy egységet alkot. (NAT 2020 II.2.1)

Az alsó tagozatban (1-4. évfolyam) a fő irányelv az egyéni különbségek kezelése, valamint a tanulási-tanítási folyamatok megismertetése. A felső tagozat (5-8. évfolyam) elsősorban a tanulási eredményességhez szükséges kulcskompetenciák kialakítására és továbbfejlesztésére összpontosít. A szakasz végére a diákok hatékonyabban fel tudják használni, esetleg alkalmazni az elsajátított kompetenciáikat az összetettebb tudástartalmak esetében. Az utolsó szakaszban (9-12. évfolyam) a diákok tudástartalmának megszilárdítása és elmélyítése a cél. Elvégzésével a tanulók az élethosszig tartó tanuláshoz szükséges kompetenciákkal rendelkezni fognak. A közoktatás kitűzött célja, hogy diákok a megszerzett tudásukat ne úgy kezeljék, mint lezárt tanulási ciklus eredményét. Az elsajátított ismeretek elősegítik a tanulók önálló tanulását. Ez megalapozza számukra a felnőtt társadalomba való sikeres beilleszkedést. (NAT 2020 II.2.1)

A programozáshoz a legtöbb műveltségi terület részben kapcsolódik. A magyar nyelv és irodalom fejezetben foglalt célok (NAT 2020: II.3.1) - például a szövegalkotás - éppen úgy elengedhetetlen a szoftverfejlesztéshez, mint az élő idegen nyelv ismerete (NAT 2020: II.3.2), hogy a szakirodalmakat önállóan fel tudja dolgozni a jövő programozója. A matematika (NAT

2020: II.3.3) és a digitális kultúra (NAT 2020: II.3.8.1) területe mutatja a legnagyobb átfedéseket a programozási tevékenység és a műveltségi területek között. A következő bekezdésekben erről a kettőről fogok bővebben írni annak ellenére, hogy a többi terület sem elhanyagolható a szoftverfejlesztéshez.

2.3. Matematikai célok a programozás kontextusában

A matematika céljait és alapelveit a NAT hat lényeges pontban foglalja össze. Az elsőként kitűzött cél a következő: „megtapasztalja a matematika értékeit, hasznosságát, szépségét". Bár elsőként szerepel a felsorolásban, lényegében a 12. évfolyam végére lehet ütemezni azt a tapasztalatot, ami a cél teljesülését elősegíti. Erre a nevelés-oktatás szakaszok szerinti bontásban utal is a programterv. Ahogy Pólya György is utal rá: „A matematika nagyon absztrakt – éppen ezért nagyon konkrétan kell előadni" (Pólya 1977: 236). Bár a konkrét példák hasznosak, azok gépies begyakorlása akadályozza a matematikai megértést. (Pólya 1977: 218-219) Mivel a problémamegoldás egy gyakorlati készség, ezért leggyakrabban utánzással tudjuk elsajátítani. (Pólya 1977: 24) Ha feltételezzük, hogy a matematikatanár kevésbé támaszkodik a sablonfeladatokra az értékelés során, akkor sem lehetünk biztosak a komplex matematikai struktúra hiánytalan kialakulásában. A matematikai problémamegoldás absztrakciója az egyik legfontosabb értéke a tantárgynak, amelyet implementálhatunk a különböző tudományokban (pl.: fizika, kémia, programozás). Ennek a magasabb fokú absztrakciónak a megértése tapasztalataim szerint nem magától értetődő, mert a matematikai tudásunkat a rendszerezett struktúra nélkül is tudjuk alkalmazni a különböző tudományterületeken.

A második meghatározott cél a NAT-ban a „megismerje a matematikai gondolkodás természetét és a matematika alapvető sajátosságait". Véleményem szerint ezt a célt egyszerűbb elérni. A matematikai gondolkodás értelmezhető úgy, hogy a definíciók között különböző relációk fedezhetőek fel, és egy matematikai objektum a tulajdonságai változtatásával egy általánosabb, vagy éppen szűkebb objektum értelmezését kapjuk (Varga 2001: 70, Varga 2001: 26). Ez a megállapítás a programozás egyik legnépszerűbb paradigmájának is az alapelve.

A harmadik meghatározott cél a NAT-ban a matematikához kapcsolódóan a következő: „fejlessze a szövegértését, a szövegalkotó és absztrakciós képességét a matematika nyelvének és szimbólumainak szóbeli és írásbeli alkalmazása során". A dolgozatom második részében az absztrakciós képesség kialakulásának segítésére teszek kísérletet. E képesség hiányában a feladat megoldása és programozási nyelvben való implementálása nem lehetséges. Egy feladat

megoldásának tervezése szükségszerűen nyelvfüggetlen általánosítás, melyet az implementáció idejében specializálunk a nyelvi sajátosságok figyelembe vételével. Pólya György erre jelenségre az igényesség paradoxonaként hivatkozik. Az állítás kimondja, hogy az általánosabb megoldást könnyebb létrehozni, mint speciálisat. (Pólya 1977: 70) A programozás esetében ez azt jelenti, hogy egy általánosabb megoldás megértésével nem kell minden egyes programnyelv esetén újra megtanulni egy algoritmust. Meglepő módon azonban a digitális kultúra online tananyaga (NKP-DK9 2020) erről nem tesz említést, amikor a programozást tárgyalja.

A negyedik célként a NAT a következő célt tűzi ki: „fejlessze a számolási készségét, a modellezési, a problémamegoldó és döntési képességét". Annak ellenére, hogy a korábbi pontok is implicit magukba foglalják a problémamegoldó képességet, ez a pont az eddigiek összefoglalására és kiegészítésére utal. Míg az előbbi célok a problémamegoldás analízis részére vonatkoznak, itt elsősorban a szintézis a hangsúlyos. Szintézis alatt azt értjük, hogy az analízis során létrehozott tervet alkalmazzuk a feladat megoldása során. (Pólya 1977: 200-205) Mivel a számolási készséget, valamint a modellezési folyamatokat a matematikához kapcsolódó példákon keresztül sajátítják el a diákok, ezért ezt a célt érdemes szorosan a tantárgyhoz kapcsolódóként értelmeznünk. Emiatt ebben az esetben nem beszélhetünk a programozásban is alkalmazható tudásról. Egy releváns kapcsolódási pont lehet az említett cél és a szoftverfejlesztés között, ha azt a magasabb szintű absztrakciót vesszük alapul, hogy az analízis-szintézis egy programozási feladat során is alkalmazható.

Az ötödik, amit a NAT megfogalmaz a matematika tanulásának legfontosabb céljaként: „fejlessze a logikus, pontos, kreatív, mérlegelő, stratégiai és rendszerező gondolkodását". Ha megfigyeljük, akkor a legtöbb modern heurisztika által definiált gondolkodási művelet (lásd: 3.1 fejezet) fellelhető a célban. A matematikai feladatok megoldásakor olyan tervezési műveletek végzünk, melyeket a korábbi feladatok során felkutatott általános vonásokkal állítunk párhuzamba. Az önálló feladatmegoldásban az így szerzett tapasztalatok könnyedén logikai modellekbe szervezhetőek, melyek segítik a későbbi ismeretlen feladatok megoldási menetének felderítését. (Pólya 1977: 177-180, Pólya 1977: 94-96)

Mivel a programozás segítségével történő problémamegoldás során is a korábbi ismereteink mozgósításával próbálunk rájönni egy feladat megoldására, ezért könnyen észrevehetjük, hogy a modern heurisztikában foglalt gondolati műveleteket nem csak a matematika területén belül hasznosíthatjuk. A matematika rendszere – legyen az algebra, számelmélet, vagy a geometria - és relációi felhasználhatóak példaként más tudományok rendszereinek értelmezésekor. Descartes így fogalmaz: „Aki figyelmesen nézi majd azt, amire gondolok, könnyen megérti, hogy éppenséggel nem a közönséges matematikával foglalkozom itt, hanem egészen más

tudományt adok elő, amelynek amazok inkább ellepleződi, mint részei" (Descartes idézi Lakatos 2021: 118) Ez alapján a matematika során felfedezett stratégiák és logikai kapcsolatok más tudományokban is hasznosíthatóak.

A kreatív gondolkodást értelmezhetjük a kreatív folyamat eredményének. Ennek a folyamatnak két különböző esetét különböztethetjük meg. Az egyik esetben a céltermék véletlen események következtében jön létre, melyet a meglévő tudásunk befolyásol. A másik irányzat szerint a cél meghatároz előre bizonyos korlátokat, amelyek az implementációt befolyásolják. A problémamegoldás esetében az utóbbi megközelítést ismerhetjük fel. Egy algoritmus implementációja közben mindkét eset valószínűsíthető, ha egy probléma megoldása során egy olyan részmegoldást implementálunk, melynek az általánosítását egy eddig megoldatlan másik feladatnál alkalmazni tudunk. (Pentelényi 2011: 22-23) Amennyiben egy probléma megoldásához segédfeladatot írunk, akkor is hasonló gondolati műveleteket végzünk. (Pólya 1977: 224-230)

A hatodik célja a Matematikának a NAT szerint az, hogy a tanuló „alkalmazható tudásra tegyen szert". A korábbi célok részben, vagy egészben a problémamegoldásról szólnak. A célok lekorlátozása csak a matematikai tartományra azonban lehetséges, hogy gátolja az általános fogalmak kialakulását, amit a diákok fel tudnak hasznosítani a programozásban is. Bár a matematikai definíciók alkalmazása hasznos lehet más tudományterülethez tartozó problémák megoldása során is, a gondolati műveletek általánosításával további előnyökre tehetünk szert. A problémamegoldás folyamatához tartozó eszközök, valamint a tervezéshez kapcsolódó módszerek megismerésével könnyebb a szintézis és analízis olyan tartományspecifikus területeken is, amelyek nem a matematika fogalmait használják.

A célok alapján a matematikai apparátus megalapozza azt a szükséges gondolkodást, amelyet a diákok a programozás során alkalmazni tudnak. A legfontosabb feladatként definiálható az, hogy az itt megismert algoritmusokat és analógiákat a tanulók más tantárgyak keretei között is alkalmazhassák. Ezáltal a matematika hasznosságára is példákat láthatnak a diákok, illetve az új ismereteik is kapcsolódni fognak a korábban megszerzett tudásanyagukhoz. A következőkben a Digitális Kultúra tantárgy NAT-ban definiált céljait elemzem, összevetve a matematikában megfogalmazott célokkal.

2.4. Digitális kultúra oktatás-nevelési szakaszai

A Digitális kultúra tantárgy célkitűzései között találkozhatunk digitális információfeldolgozással kapcsolatos, informatikát részben érintő, valamint programozáshoz és problémamegoldáshoz kapcsolódó előírásokkal is. A programozáshoz szükséges a digitális forrásanyagok önálló feldolgozása, és a számítástechnikai környezet által biztosított eszközök működésének ismerete is. A kooperációs eszközök és az online eszközök eredményes használata elengedhetetlen egy szoftverfejlesztő számára, de megkönnyíti a diákok információs társadalomba való beilleszkedését is. A tantárgy célkitűzéseinek ismertetésekor az általános célok helyett, a nevelés-oktatási szakaszokhoz tartozókat fogom elemezni. Ennek az az oka, hogy a matematika tanulmányok során elsajátított, problémamegoldáshoz szükséges gondolati műveletek hasznosítási lehetőségeire szeretnék rámutatni. Ezeket szűkíthetjük az életkori sajátosságok figyelembevételével, így körvonalazódhat előttünk a matematika-problémamegoldás-programozás közötti kapcsolat.

A Digitális kultúra tantárgyat 3. osztálytól tanulják a diákok. A 8-10 éves korban lévő tanulók számára kitűzött cél a digitális eszközök használatának megismertetése. Ez a Piaget szerinti kognitív fejlődési szakaszokkal is indokolható. A tárgyak logikai rendszerbe illesztése a konkrét műveleti szakasz végére alakulhat ki, mely a 11. életév környékén esedékes. Természetesen ettől eltérhetnek egyéni esetekben a tanulók. (Atkinson és mtsai 2005: 100 -102) Emiatt azonban a matematika alkalmazása digitális környezetben az első oktatás-nevelési szakaszban nem szerepel, mint kitűzött cél.

A második oktatás-nevelési szakaszban a tanulók számára kitűzött cél, hogy megismerkedjenek a szintézis egy új formájával. A NAT így fogalmaz: „Az algoritmizálás, programozás ismerete elősegíti az olyan készségek fejlesztését, amelyek a problémamegoldásban, a kreativitás kibontakozásában és a logikus gondolkodásban nélkülözhetetlenek.” A korábban említett életkori sajátosságok miatt feltételezhető, hogy a matematika tantárgy keretei között megismerhető analízis-szintézis folyamattal párhuzamosan ismerkednek meg a digitális környezetben való implementációval is. A programozás és a problémamegoldás itt még nem feltétlenül kapcsolódik össze az elérhető online tankönyvek szerint (NKP-DK5 2020, NKP-DK6 2020), sokkal inkább a pozitív élmények és az érdeklődés felkeltése a cél. Annak ellenére, hogy nem azonosítható a szoftverfejlesztéssel az 5-6. évfolyamon szerzett tapasztalat, a tanulók az elsajátított fogalmakkal lehetőséget kapnak a megtervezett algoritmusaik implementálására egy matematikától eltérő nyelvi környezetben.

Ugyancsak a második oktatás-nevelési szakasz kitűzött céljaként található a NAT-ban a következő: „A tanuló a digitális kompetenciák és a problémamegoldás képességének fejlesztése során... problémaorientált feladatmegoldási módszereket sajátít el” Bár a Nemzeti Közoktatási Platform 2020-as NAT-hoz készült tankönyve 7.-8. évfolyamra még nem készült el, a kitűzött célból következtethetünk arra, hogy más tantárgyakhoz kapcsolódó (pl.: fizika, kémia) problémákat implementálnak digitális környezetben a tanulók. A matematika tantárgyat követve, a tervezési képesség fejlődésével párhuzamosan, újabb implementálási módokat ismernek meg a diákok. Felső tagozatban így a diákok nem csak a Természettudomány és földrajz műveltségi területen belül, de a digitális technológiákban is hasznosíthatják matematikai tudásukat.

A harmadik oktatás-nevelési szakaszban a Digitális Kultúra tantárgy céljai között arra találunk utalásokat, hogy a programozási eszköztár bővítése kerül a fókuszba. Erről tesz említést a következő célkitűzés a NAT-ból: „A problémák összetettségében építeni kell a korosztályra jellemző, magasabb absztrakciós szintre, így a korábban elsajátított ismeretek bővítése nemcsak konkrét új fogalmak bevezetését, hanem az ismeretek felhasználási területének bővítését is jelenti.” A tanulók az elsajátított matematikai tudásuk segítségével a programozásban különböző struktúrákat valósítanak meg. A komplex struktúrák absztrakciójára a matematikában számtalan példa van, többek között a paralelogramma (Varga 2001: 314-315). Ezek alapján a programozási nyelvekhez tartalmazó fogalmakat is bővíteni tudják a diákok. A digitális környezetben létrehozott implementációt változatosabbá és olvashatóbbá teszik ezáltal a diákok, illetve a problémák különböző megoldását is gyakorolhatják, ami segíti az absztrakciók értelmezését.

Bár a digitális kultúra 11. évfolyamos tankönyve egyelőre nem elérhető, a következő célkitűzésből azonban következtethetünk annak tartalmára: „Az algoritmizálásnál a hétköznapi feladatok mellett a más tantárgyakban megjelenő folyamatok modellezése, vizsgálata továbbra is olyan cél, amellyel a tantárgyi koncentráció erősíthető, és bemutatható a programozás ilyen irányú hasznosításának lehetősége. A programozás fogalmainak ismeretét a tanuló magas szintű, széles körben elterjedt, de egyszerű formális programozási nyelv segítségével mélyíti el.” Míg a 9-10. osztályos tankönyv (NKP-DK9 2020, NKP-DK10 2020) a sablonfeladatokra építi a programozás szintaktikájának megismerését, feltehetően a 11. osztályos a problémamegoldást helyezi majd a fókuszba, hogy a diákok a matematikában elsajátított módszereket digitális környezetben is tudják hasznosítani. A hétköznapi problémák megoldásakor fontos az itt megszerzett tapasztalat, melynek során el tudnak majd vonatkoztatni a matematika világától.

2.5. Összegzés

Ahogy láttuk a NAT kitűzött céljainak elemzésekor, a Digitális Kultúra és a Matematika egymással párhuzamosan haladnak a különböző oktatás-nevelési szakaszokon keresztül. Bár vannak ütközési pontok a két tantárgy között, azok korántsem mindig triviálisak, ha a matematikai struktúrákkal, és az azok közötti relációkkal nem vagyunk tisztában. A matematikai fogalmak felhasználásával teszek kísérletet arra, hogy általánosan bemutassam az analízis folyamatát. A folyamat megértésének segítségével az algoritmusok, és képletek szóról szóra történő betanulása elkerülhető lehet.

A matematika tartományától való eltávolodás, mint a probléma megoldásának egy módszere, nem újszerű gondolat. Descartes azért tanulmányozta a geometria, az analízis és az algebra területét, hogy a három terület ötvözésével egy olyan absztrakciót kapjon, melyet más tudományokra is ki tud terjeszteni. Célkitűzése az volt, hogy gondolatait úgy rendszerezze, hogy az általa „feleslegesnek" vagy „fárasztónak" ítélt matematikai területeket elhagyja. Ezáltal úgy vélte, hogy tartományfüggetlenül tudja majd tanulmányozni a különböző tudományágakat. (Lakatos 2021: 118-120)

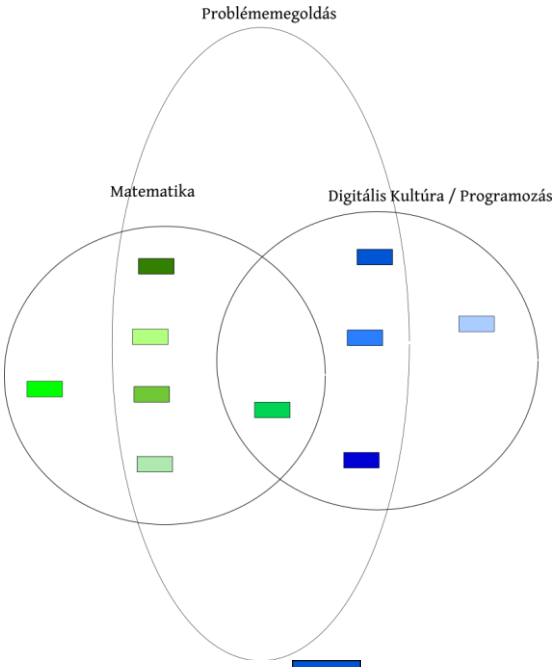
Pólya problémamegoldásra definiált lépéseit bár matematikai példákon vezeti le, a szakkifejezések kiküszöbölésének fontosságára hívja fel a figyelmet abban az esetben, ha ismeretlen feladattal találkozunk. (Pólya 1977: 80) A 4 általános feladatmegoldási lépésből csak egy szól az implementációról, melyet a terv tökéletesítése előz meg. Mivel a feladatmegoldás során minden lépésnél a korábbi tapasztalatainkra építünk, ezért alkalmazható általánosan az eredetileg matematikai problémákra alkotott feladatmegoldási algoritmus. (Pólya 1977: 53-55, Pólya 1977: 137-139) Hasonlóan Descartes-hoz Pólya is a gondolatok rendszerezésének fontosságát emeli ki, mely a problémamegoldás során a leghasznosabb. A matematika hasznossága, hogy ilyen rendszerekre természeténél fogva példákat ad. (Pólya 1977: 171-173)

Dienes a formális matematika tanításának reformjában látta a tantárgy oktatásának fejlesztési lehetőségét. Pólyához hasonlóan úgy gondolta, hogy a tantárgy célja a felismerés, amelyet a tartományfüggetlen segédeszközökkel könnyebb elérni, mint képletekkel. Módszertan szempontjából ez azt jelenti, hogy olyan tapasztalatokat kell átadni a diákoknak, melyekkel könnyebben fel lehet építeni a fogalmakat. Ezek lehetnek hétköznapi életből merített szituációk, de akár mesterségesen létrehozott helyzetek is, melyeket tartományfüggetlen elemekkel egészítünk ki. Amennyiben a célunk a felismerés élményének átadása, amit a

problémamegoldásban is hasznosítani tudnak a diákok, akkor a tanár csak mint útmutató vesz részt a tanítás-tanulás folyamatában. (Dienes 2015: 31-34)

Az említettek tükrében az algoritmizálás készségét a problémák megoldásának tervezésében látom hasznosnak. Legyen az matematikai probléma megoldása, vagy programozási feladat implementálása, a tervezési fázisban történő gondolati műveletek szükségszerűen tartományfüggetlenek. Ez nem azt jelenti természetesen, hogy nem kell tartományspecifikus tudással rendelkeznünk egy probléma megoldásához. Egy geometriai probléma megoldásához tudnunk kell néhány definíciót a feladat leírásának értelmezéséhez, éppen úgy, mint egy programozási feladathoz. Ha egy probléma specifikációját a megoldás első lépéseiben a szakkifejezések megszűrésével egyszerűsítjük, akkor tartományfüggetlen problémamegoldást kapunk. Ezek után érdemes a gondolati műveleteket ebben a független tartományban végezni az implementációig. Dolgozatomban a feladatok megoldásait ezért nem implementálom pszeudokódban, és nem írok fel matematikai képleteket sem.

A következő ábra tartalmazza a NAT-ból idézett célokat az alapján rendszerezve, hogy azok mennyire törekednek az problémamegoldás absztrakt módszereinek ismertetésére.



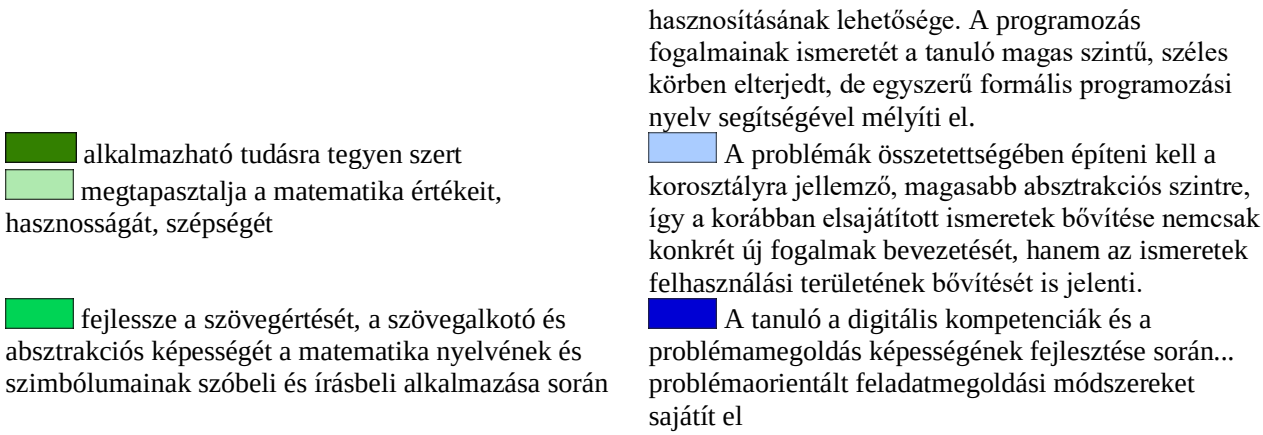
megismerje a matematikai gondolkodás természetét és a matematika alapvető sajátosságait

fejlessze a logikus, pontos, kreatív, mérlegelő, stratégiai és rendszerező gondolkodását

fejlessze a számolási készségét, a modellezési, a problémamegoldó és döntési képességét

Az algoritmizálás, programozás ismerete elősegíti az olyan készségek fejlesztését, amelyek a problémamegoldásban, a kreativitás kibontakozásában és a logikus gondolkodásban nélkülözhetetlenek.

Az algoritmizálásnál a hétköznapi feladatok mellett a más tantárgyakban megjelenő folyamatok modellezése, vizsgálata továbbra is olyan cél, amellyel a tantárgyi koncentráció erősíthető, és bemutatható a programozás ilyen irányú



1. ábra: A problémamegoldás, a matematika és digitális kultúra/programozás kapcsolata (Forrás: Saját készítés)

Az összesítő ábrán látható, hogy a NAT-ban kitűzött célok között találhatunk olyanokat, melyek tartományspecifikusak (,). Ezeket jellemezhetjük úgy, hogy az implementációt befolyásoló tapasztalatok összessége. Ilyen lehet például a Python egy beépített függvénye, vagy matematikai műveletek ismerete. Azok az elemek, amik az adott tantárgy halmazába, de nem a metszetbe tartoznak (, , , , , ,), a tervezést meghatározó tantárgyfüggő elemek. Ezek közé tartozik párhuzamosság definíciója, az OOP osztályszerkezetek meghatározásának menete, vagy a *nevezetes algoritmusok* implementációja. A három halmaz közös metszetében ( ) azt az elemet láthatjuk, ami a tartományfüggetlen tudásra utal. Az a cél tartalmazza többek között azt a készséget is, hogy a korábbi feladatmegoldásaink tapasztalatának felhasználásával képesek leszünk ismeretlen feladatok megoldását is könnyíteni. A dolgozatomban egy példát szeretnék ismertetni, amely segítséget nyújthat ennek a célnak az elérésében.

3. A programozás és a matematika közötti kapcsolat

A problémamegoldás képessége különbözteti meg leginkább az emberiséget az állatvilágtól (Pólya 2010: 9). E tevékenységhez kapcsolódó gondolkodási műveletekre szükségünk van a hétköznapi életben (Molnár – Pásztor-Kovács 2015: 341) éppen úgy, mint a természettudományok vagy a programozás cselekménye közben (Szlávi - Zsakó 2016:2). Bár a matematika elsősorban a problémamegoldásról szól, a tapasztalatok azt mutatják, hogy az egyetemre jelentkezők nem rendelkeznek a szükséges ismeretekkel (Tasnádi 2017). Személyes élményeim alapján a problémaosztályok fogalma először a programozás tanulmányaim során került elő, mert a matematika tantárgy keretei között a típusfeladatok mechanikus begyakorlására ment el az idő. Célom az, hogy bizonyítást nyerjen az a feltételezésem, hogy a matematika előnyeit kihasználhatjuk a programozás oktatása közben.

A problémamegoldás oktatását érdemes megvizsgálni pszichológiai és logikai oldalról egyaránt. Ha a problémamegoldást pusztán logikai oldalról szemléljük, elkövethetjük azt a hibát, hogy célunk nem az eljárás megértésének segítése, hanem a tanulók meggyőzése arról, hogy helyes a megoldásunk. Ezzel szemben, a pszichológiai oldalról akkor beszélünk, ha magát a megértést vizsgáljuk. Ha a két oldalt összemossuk, akkor nehéz megkülönböztetni, hogy matematikai gondolatokat tanítunk, vagy matematikai megértést. A gondolatok átadása és a megértés segítése között minőségbeli különbségeket fedezhetünk fel (Skemp 2005: 17-19).

A matematika fontosságát azért hangsúlyozom, mert a problémamegoldási műveleteket, melyeket programozás közben alkalmazunk, e tantárgy keretében tudjuk egyszerűen megérteni. René Descartes azt mondta, hogy minden problémát vissza lehet vezetni matematikai problémára. Pólya György utal rá, hogy Descartes optimizmusa kicsit indokolatlan, de vannak bizonyos stratégiák arra vonatkozóan, hogy hogyan is tudunk nekiállni egy probléma megoldásának (Pólya 2010: 12).

Kutatásomban a matematika logikáját igyekszem felhasználni ahhoz, hogy a problémamegoldás heurisztikus rendszerében (Pólya 2010: 10) könnyebben tájékozódjunk. A feladatok megoldásával nem az a célom, hogy egy minden esetre alkalmazható módszert bevezessek, sokkal inkább problémák mögött rejlő matematikai struktúrákra szeretném felhívni a figyelmet.

A programozás és a matematika technikai kapcsolatára Alan Turing hívja fel a figyelmet. A matematikus tézise még az előtt kimondta, hogy minden komplex folyamat lefordítható véges számú aritmetikai és logikai műveletekre, mielőtt a számítógépek megjelentek volna.

Megfigyelhető az a jelenség, hogy számos esetben az elmélet kimondása megelőzte a műveletek megvalósítására képes környezetek megjelenését. Neumann Jánost többek között Turing tézisei irányították, miközben a fizikai gépeket megvalósította. (Pléh 2012: 186-187)
Az említett jelenség oktatással való párhuzama abban figyelhető meg, hogy a matematikai apparátus elsajátítása megelőzi a programozási ismeretek megszerzését. Emiatt lehetséges a program fejlesztésének menetét a matematikai fogalmak felhasználásával segíteni. A logikai és matematikai modellek ismertetésével egy olyan absztrakciót biztosíthatunk a diákok számára, amelyet a szoftverfejlesztés során alkalmazni tudnak.
Jelen munkámban kísérletet teszek arra vonatkozóan, hogy problémamegoldásra témaköréből kiemelt feladatokat szemléltetek halmazműveletek segítségével. Választásom azért esett a halmazműveletekre, hogy valósággal könnyen párhuzamot tudjunk vonni (Skemp 2005: 193-195). A probléma megoldása közben segítségünkre lehet, hogy felfedezzük az adott problémaosztályba tartozó feladatokat, ezzel segítve a fogalomalkotás folyamatát (Skemp 2005: 109).

3.1. A gondolkodási műveletek

A gondolkodási műveletek pszichológiai vizsgálatakor érdemes megismerkednünk az intelligencia fogalmával. Hebb az intelligenciát két részre osztotta. Az egyik az úgynevezett A intelligencia, amely a fejlődési képességet jelenti, míg a B intelligencia a felfogóképességet foglalja magában. Míg az előbbi többnyire velünk született képességeket tartalmaz, utóbbi felfogható az intelligencia használatának, melynek fejlődése a környezetünk megértésén alapul, így matematikai tudásunkkal egyértelműen kapcsolatba hozható B intelligenciánk fejlődése (Skemp 2005: 21-23).
A gondolkodási műveletek közben segítségünkre van az úgy nevezett fogalmi gondolkodás. E tevékenység szerepe akkor értékelődik fel igazán, ha figyelembe vesszük Mérő László által megfogalmazott pszichológiai korlátját az emberi elmének, miszerint a „gondolkodási struktúrák maximális száma egyénenként 7 ± 2 érték mögött mozog” (Szlávi - Zsakó, 2016: 5). A fogalmi gondolkodás legnagyobb ereje abban rejlik, hogy tapasztalatainkat különböző osztályokba tudjuk rendezni és ezen osztályok között tudunk kapcsolatokat létrehozni, a speciális esetek helyett[1]. Minél magasabb szintű absztrakciókat tudunk felépíteni, annál több

1 Véleményem szerint az objektumorientált programozásban kiemelt jelentőségű ez a készség. Elég csak a tervezési mintákra gondolni. A tervezési minták olyan fogalmak, melyek több esetet fednek le, és a programozó dönti el, hogy az adott esetben melyik tervezési mintára van szüksége. Hibás fogalmi gondolkodás esetén rossz mintára eshet a választásunk.

esetre tudjuk azokat alkalmazni, így a pszichológiai korlátot „megkerülhetjük”. A matematikában rejlő lehetőség itt a legszembetűnőbb, hiszen a legelvontabb rendszerről beszélünk (Skemp 2005: 40-41). Bár magam is úgy gondoltam, hogy a matematika a szabályrendszerek tanulásáról szól, valójában a hangsúly a fogalmi gondolkodás kialakításán van, hisz ez az a tudás, amelyet a legtöbb tudományban hasznosítani tudunk.

Amennyiben már több fogalommal rendelkezünk egy adott tevékenységgel kapcsolatban, azok szintén struktúrákat alkotnak, amelyeket *szkémáknak* nevezünk. A *szkémák* kialakítása azért fontos, mert ezzel a fogalmaink között biztosítjuk a kapcsolatokat. Ezen szellemi struktúrák felépítése csakis értelmes tanulással történhet, mivel a fogalmak csak akkor rendeződhetnek magasabb szintű struktúrákba, ha azokat már korábban megértettük. A *szkémák* megkönnyítik a fogalmak rendszerezését, és korábbi fogalmakhoz is kapcsolódhatnak[2]. Mivel tudásunkat nagy mértékben befolyásolják ezen struktúrák közötti kapcsolatok, elmondhatjuk, hogy minél több *szkémával* rendelkezünk, annál biztosabban találunk megoldást egy speciális problémára. Ha figyelembe vesszük, hogy ezek a struktúrák a fogalmak általánosításai, két hatalmas előnyt emelhetünk ki: az új fogalmak tanulása megszilárdíthatja a már meglévő *szkémáinkat*, valamint tovább csökkentik az egyidejű terhelését elménknek, melynek korlátait már a korábbiakban definiáltuk (Skemp 2005:53-58).

A *szkémák* bár segítenek az új ismeretek elsajátításában, egyben gátolják is azt. A problémamegoldás esetén a kialakult szellemi struktúrák rugalmassága határozza meg, hogy az új tapasztalatokat mennyire tudjuk asszimilálni a már meglévő *szkémáinkba*, valamint szükség esetén milyen szintű akkomodáció megy végbe. Ha az előbb említett folyamatok nem zárulnak le sikeresen, a későbbi tapasztalataink integrálása nehézségekbe ütközhet. Míg az asszimiláció az új fogalom struktúrába illesztését jelenti, az akkomodáció a meglévő struktúra átalakítására vonatkozik (Skemp 2005:53-58).

Azt a jelenséget, amikor egy új tapasztalat segítségével felismerünk egy új összefüggést, és sikeres asszimilációt hajtunk végre egy már meglévő *szkémánkkal*, *insight* jelenségnek nevezzük. Ez a cselekmény nem tudatos (Skemp 2005:73). Amennyiben megtörténik az *insight*, az hatással van a későbbi eredményeinkre is, aminek következtében várhatóan eredményesebben küzdjük le a hasonló felépítésű feladatokat (Klein – Greer – Zétényi 2020:143-151).

Az ebben a részben tárgyalt gondolkodási műveletek fontosságát jól mutatják, hogy az informatikai gondolkodást a következőkben definiálták a közelmúltban: minta-felismerés,

2 Az objektumorientált programozásban is gyakori művelet. A SOLID elvek betartása során hasonló struktúrák kialakítása a cél.

elemi részekre bontás, az absztrakciók megértése, algoritmusok létrehozása és használata (Csernai 2020: 5).

3.2. Motiváció

Csakúgy mint a matematikában, a programozásban is fontos a motiváció fenntartása a diákokban. A problémamegoldásban még kezdő tanulók gyakran találkoznak megoldhatatlannak tűnő problémákkal, ezért a motiváció kialakítása elengedhetetlen, hogy a kezdeti akadályok leküzdése közben ne a sikertelenség érzése domináljon, ami a csökkenő érdeklődésnek egy kiváltó oka (Pásztor 2014).

A motivációk két csoportját különböztethetjük meg az alapján, hogy honnan érkezik a motívum. Eszerint definiálhatunk intrinzik és extrinzik motivációs tényezőket. Míg az előbbi motívumok esetén arról beszélünk, hogy a tevékenység önmaga a jutalom, addig az utóbbi esetén a jutalom a külső környezetből jön (Józsa 2002a: 3). Az extrinzik motiváció tipikus példája a jutalmazás és büntetés, amellyel az oktatás esetében is találkozhatunk. Pedagógusként törekednünk kell arra, hogy támogassuk az intrinzik motiváció kialakulását a tanulás folyamatában (Pajor 2015: 25).

Amennyiben a tanítás céljaként az önszabályozó tanulást fogalmazzuk meg, akkor figyelembe kell vennünk a tanulók fejlesztendő tanulási motívumait. Ezeket a motívumokat befolyásolják többek között a kulturális hatások, a korábbi tanulási tapasztalatok, valamint a szociális környezet (Réthy 2002: 10). A pedagógus szerepe ezeken a területeken megkérdőjelezhetetlen, hiszen mintaként szolgál a diákjai számára, ám munkáját nehezítik egyrészt a tanulók között található kompetenciabeli különbségek (Józsa 2002a: 16), másrészt a diákok eltérő önképe változtatja a viszonyát az adott tantárgyhoz (Gaskó 2006: 32).

A célorientációs elméletek alapján három csoportot definiálhatunk az alapján, hogy mi az elvárt megítélés és milyen eredménycélt tart fontosnak a tanuló (Kő – Pajor – Szabó 2017: 319). Ezek alapján megkülönböztetünk elsajátítási célokat, közelítő-viszonyító célokat és elkerülő-viszonyító célokat (Pajor 2015: 43).

A viszonyító célok esetében dominál, hogy a célt a környezet definiálja (Kő – Pajor – Szabó 2017: 321), míg az elsajátítási célnál a fókusz az elsajátítandó tananyagon van (Pajor 2015: 43), tehát a tanuló elsősorban ismeretei elmélyítésére koncentrál.

Közelítő-viszonyító teljesítménycél esetében az a motiváció, hogy a társas környezet teljesítményén túlteljesítsen a tanuló. Ennek egyértelmű hátránya, hogy a túlteljesítési kényszer a tudásanyagban való elmélyülés rovására megy.

Elkerülő-viszonyító teljesítménycélról akkor beszélünk, ha a tanulót az motiválja, hogy elrejtse esetleges inkompetenciáit a külvilág elől. Ebben az esetben sajnos nem eredményes a tanuló, mivel csak a minimumot szeretné elérni, így tudás elmélyítéséről az esetek többségében nem beszélhetünk (Pajor 2015: 43).

Az elsajátítási céloknál, ahogy azt korábban már említettük, a fő hangsúly a tudás elmélyítésén van. Megvizsgálva a többi teljesítménycélt megállapíthatjuk, hogy ebben az esetben beszélhetünk leginkább az önfejlesztés elősegítéséről. Ezen célok által motivált tanuló esetén nem az eredmények produkálása a cél, sokkal inkább adott készség megfelelő szintű elsajátítása (Józsa 2002b: 81). Pedagógiai szempontból tehát törekednünk kell egy olyan környezet kialakítására, ahol a tanuló számára világossá válik, hogy e célok nem elérhetetlenek (Kő – Pajor – Szabó 2017: 333-335).

Ha összehasonlítjuk az elsajátítási és a viszonyító célokat, megfigyelhetünk további különbségeket is a két célstruktúra között. Egyik legszembetűnőbb, hogy a közelítő-viszonyító céllal rendelkezők jobban teljesítenek, mint elsajátító célok által motivált tanulók, ha az elvárt jutalmazás a teljesítmény alapján történik (Pajor 2015: 43). Ez megerősíti a következtetést, hogy a célokat, melyekre belső motivációként tekintünk, a környezet befolyásolja. Ugyancsak megfigyelhető, hogy a tanári-diák kapcsolat, beleértve a tanár felől érkező érzelmi támogatást is, befolyásolja az elsajátítási célok kialakulását (Molnár – Fodor – Kurucz 2020: 35).

A motiváció kialakításában a tanár személyének is meghatározó szerepe van az általa oktatott tantárgyak esetén (Molnár – Fodor – Kurucz 2020: 35). A megfelelő kapcsolat a tanulókkal azért is fontos, mert csökkentheti a szorongás mértékét. Az említett jelenség akkor alakulhat ki, amikor tanulók között megértésbeli különbségek alakulnak ki. A szorongás egy darabig magasabb teljesítményre is ösztönzi a szorongó diákot, ám amikor látja, hogy erőfeszítései hiábavalóak, egyre kevesebb energiát fektet az anyag megértésébe, mert azt gondolja, hogy felesleges. Attól a ponttól kezdve, hogy a tanulóban kialakult az a kép, hogy ő úgysem érti, egyfajta önbeteljesítő jóslatként fog funkcionálni a saját megállapítása (Skemp 2005: 170-171).

3.3. Személyiségfejlődés

Az előző fejezetben már esett szó a pedagógusszerepről, ami a problémamegoldás oktatása során kiemelt jelentőségű. Ebben a részben a pedagógusmintát és a személyiségre gyakorolt hatásait vizsgálom.

Abban az esetben, amikor az oktató az eljárások begyakorlására helyezi a hangsúlyt, a tanulók számára nem válik egyértelművé azoknak a gyakorlati jelentősége. Amennyiben az említett eljárások megértésére nem fordítunk figyelmet, azok nem járulnak hozzá a diákok személyiségfejlődéséhez. Személyiségfejlődés alatt azt a pszichológiai folyamatot értjük, amikor a tapasztalatokat személyiségünkbe integráljuk. Akkor sikeres a személyiségfejlődés, ha végbemegy az integráció és az integrált személy többek között konstruktívan cselekszik adott problémahelyzetben ahelyett, hogy bírálná azt. Ha a személyiségfejlődés szempontjából nézzük a tanulók matematika tanulását, valószínűleg megállapítható, hogy amikor nincs kellő motiváció és törekvés a matematika rendszerszintű megértésére, akkor nem beszélhetünk integrációról. Ha a motiváció hiánya fennáll, akkor a tanuló elutasítóvá válhat a tevékenységgel szemben, ezáltal nem válik a személyisége részévé megszerzett tapasztalat (Dienes 2015: 29-31).

Gyakori érvelés, hogy a matematika fegyelemre tanítja a diákokat, és tudni kell, nem pedig érteni. Ezen érveléssel az a probléma, hogy magában hordozza azt a gondolatot, hogy a problémamegoldás képességét nem kell személyiségünk részéve tennünk, elég csak, ha amit mondanak megcsináljuk. Implicit tartalmazza azt a kijelentést is, hogy az önfegyelem kevésbé elvárt, mint a külső fegyelem. Ezt a gondolatot ha társadalmi szempontból vizsgáljuk, lehetséges, hogy nem a legelőnyösebb személyiségeket fejlesztjük (Dienes 2015: 30).

További probléma a problémamegoldási eljárások megfelelő magyarázat nélküli gyakoroltatásával, hogy akkor a jobb képességű tanulókban is a már meglévő gondolkodási struktúrák roncsolódásához vezethet. Ilyen esetekben az értelem megsértéséről beszélhetünk, mivel a tanár a számonkérésével olyan szabály alkalmazását várja el, amely a tanuló számára értelmezhetetlen (Skemp 2005: 159-160).

Ez alapján azt gondolom, hogy az absztrakciók és tételek kimondása a magyarázat nélkül nem csak nem segíti, de egyenesen gátolja a megértés folyamatát. A megértéshez szükséges tanítandó elemeket Richard R. Skemp a következőképpen foglalja össze: jól strukturált alapok, rendszerek felismerésének készsége, *szkémák* fejlesztése, ha az szükségessé válik (Skemp 2005: 73).

3.4. Szimbolika

Az eddigi áttekintés során a fogalom különböző struktúráiról volt szó. Láthattuk, hogy a fogalom alkotása és a matematikai ismeretek együtt járnak. A matematika nagy előnyét, az absztrakt és kidolgozott rendszert azonban eddig csupán elméleti szinten értelmeztük. A hiányzó elem, a szimbolika adja meg számunkra a legszembetűnőbb különbséget matematika és a programozás között.

A legtöbb problémamegoldáshoz kapcsolódó fogalom és eljárás gondolati szinten létezik csupán. Ahhoz, hogy valakivel közölni tudjuk gondolatainkat, szükségünk van egy közös szimbólumrendszerre. Ha a szimbólumok halmazát nézzük, ezek lehetnek akár természetes nyelvek, akár a matematika nyelve, vagy éppen a programozási nyelvek. Amikor kommunikálunk, akkor az a célunk, hogy szimbólumaink ugyanazt a fogalmat hozzák létre a befogadó oldalon, mint amit mi küldőként a szimbólumhoz társítottunk (Skemp 2005: 95-99).

A problémamegoldásban és a fogalomalkotásban a szimbólumrendszereknek megkérdőjelezhetetlen szerepe van. Egy feladat kezdeti szakaszában szimbólumok segítségével vizsgáljuk a problémához tartozó eseteket, így próbáljuk rendszerezni gondolatainkat. A már korábban említett *insight* ezen vizsgálatok közben jöhet létre, mely a szimbólumokat automatikusan a fogalomhoz rendeli. Sajnos ez a fogalomalkotás nem mindig helyes, és ez csak az igazolás fázisában derül ki. Ilyenkor mindig vizsgálni kell, hogy megfelelő szimbólumrendszert választottunk-e (Skemp 2005: 129).

A helyes szimbólumok kiválasztásához ismernünk kell a problémát. Egy problémának mindenképpen kell tartalmaznia legalább három információt. Kell lennie egy ismeretlennek, amit keresünk, ha ez nincs megadva, nem beszélhetünk feladatról. Tartalmaznia kell adatokat, amikből ki tudjuk következtetni az ismeretlent. Legutolsó sorban kell lennie feltételnek, ami meghatározza, hogy az adatokból hogyan tudunk következtetni az ismeretlenre (Pólya 2010: 19).

A problémamegoldás egyik lehetséges első lépése egy úgynevezett „absztrakt platform” kiválasztása, amelyen a probléma egyes elemeinek kapcsolata világosabbá válik számunkra (Kátai 2020: 12). A feladatok megoldásához én is ezt a módszert alkalmaznom, hogy bizonyítsam a matematikai fogalomalkotás és a programozás közötti kapcsolatot.

3.5. Feladatmegoldás halmazok segítségével

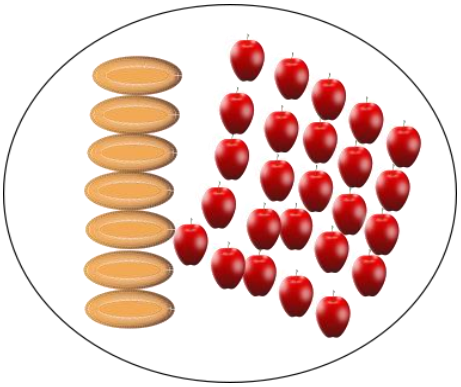
3.5.1. Maradékos osztás

Matematikai definíció: A maradékos osztás az egész számok körében értelmezendő. Mivel az egész számok halmazán nem tudunk bármely számot bármely számmal osztani, ezért maradék képződik.
Formális leírás: $a = b \cdot q + r$ és $0 \leq r \leq |b|$
Programkód: $a\%b = r$

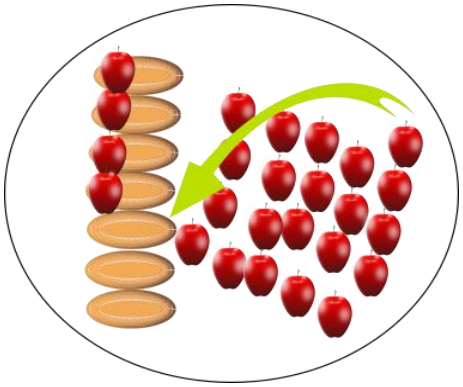
Példa: Van 25 almánk, amit 7 kosárba szeretnénk szétválogatni úgy, hogy minden kosárban egyenlő számú alma legyen, és nem vághatunk almát több részre. Mennyi almát nem tudunk elhelyezni a kosarakba?

Halmazművelet: Adott egy tetszőleges elemszámú (a) halmaz. Első lépésként bontsuk részhalmazokra úgy, hogy minden részhalmaz azonos számosságú (b). Utolsó lépésként figyeljük meg, hogy van-e olyan részhalmaz, amelynek számossága nem éri el b-t. Ha van ilyen halmaz, az lesz a maradék (r).

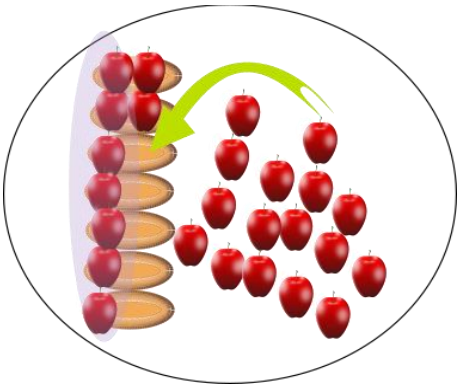
Halmazművelet magyarázata: Látható, hogy a számok megfelelnek a halmazok számosságának. Tetszőleges egész számmal megismételhető a művelet. A példában szereplő 25 alma mindegyike a halmaz egy eleme, amiben a részhalmazokat létrehozzuk. Miután csináltunk 7 elemű részhalmazokat, megmaradt az a szám, amelyet már nem tudunk kosarakba rendezni. Bár az eredményt megkaptuk, a halmazokon van lehetőségünk bejelölni a kosarakat is. Ezek olyan részhalmazok, melyek mindegyik részhalmazból 1 elemmel rendelkeznek.



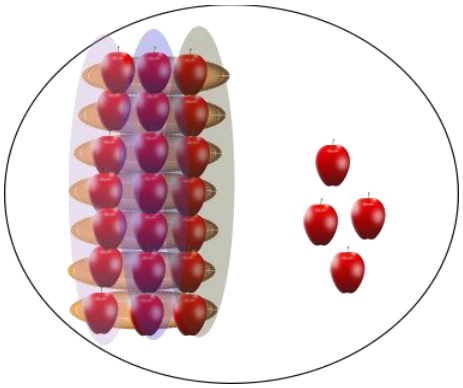
0. Kiinduló halmaz.



1. Felülről lefelé haladva helyezzünk el egy almát minden kosárba.



2. Ha már minden kosárban azonos számú alma van, akkor újra az első kosártól kezdve rakjuk az almákat.



3. Folytatjuk egészen addig, amíg már több kosár van, mint alma. A maradékos osztás eredménye azoknak az almáknak a száma melyek nem kerültek kosárba. A feladat során 3 darab 7 elemű halmaz keletkezett.

2. ábra: Maradékos osztás halmazokkal (Forrás: Saját készítés)

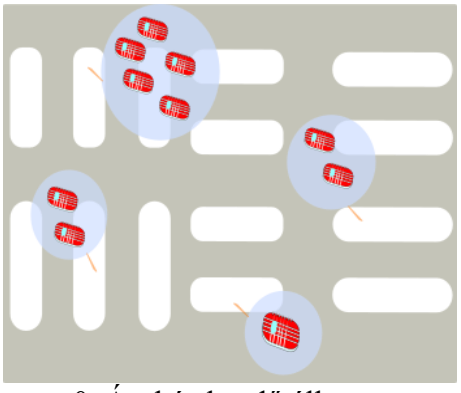
A maradékos osztás alapművelet a matematikában és a programozásban egyaránt. Megfigyelhetjük, hogy szimbolikában nem sokban tér el a programkód a formális leírásról. Felvezetését azért tartottam fontosnak, hogy a későbbiekben erre a fogalomra tudjunk építeni.

3.5.2. Tömb elemeinek összessége

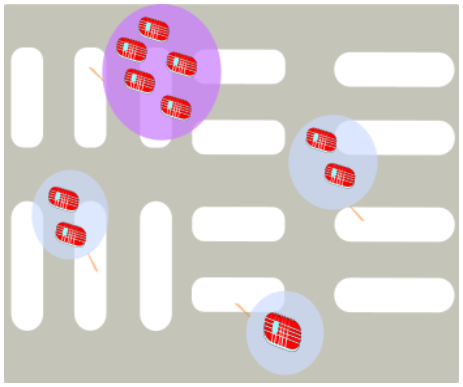
Feladat: Adott egy tetszőleges (N) elemszámú tömb. Számoljuk ki a tömb elemeinek összegét.
Példa: Egy áruházban leltározzák a négyzetrácsos füzeteket. Mivel az áruház hatalmas, ezért több helyen is ki volt rakva a termék a polcokra. Mennyi négyzetrácsos füzet van összesen?

Halmazművelet: Adott N darab tetszőleges számosságú halmaz. Annak a halmaznak keressük a számosságát, amely ezt az N darab részhalmazt magában foglalja. Vizsgáljuk meg egyesével a halmazokat, és adjuk hozzá számosságukat az eddigi részeredményekhez.

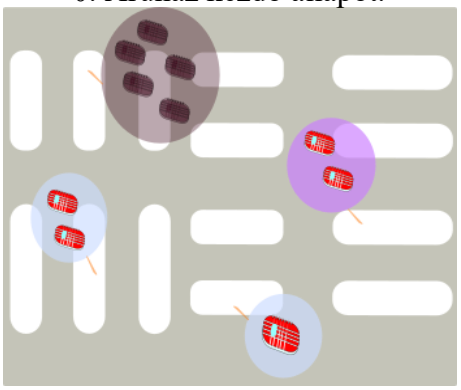
Halmazművelet magyarázata: A példánkban annyi részhalmazunk lesz, ahány helyre ki vannak téve a füzetek. Az egyes részhalmazok elemeit egyesével megszámoljuk és összegezzük.



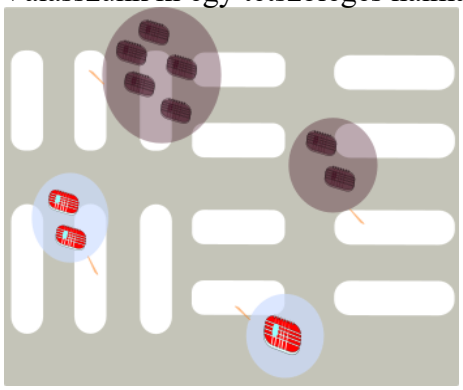
0. Áruház kezdő állapot.



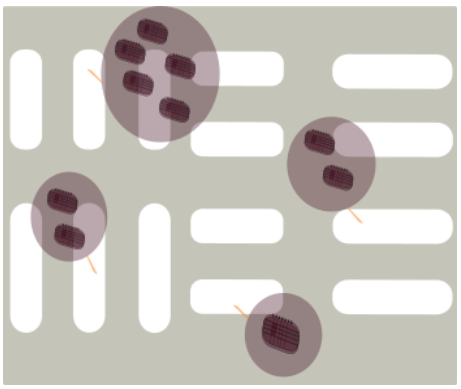
1. Válasszunk ki egy tetszőleges halmazt



2. Miután megállapítottuk a halmaz elemeinek számát (5), válasszunk egyet a megmaradt halmazok közül.



3. A választott halmaz elemeinek számát (2) adjuk hozzá az eddig számolt füzetek számához (5), majd válasszunk egy újabb halmazt. Az összeszámolt füzetek száma már 7.



4. Addig ismételjük a korábbi lépéseket, míg végül már nem tudunk újabb halmazt választani. Miután hozzáadtuk az eddig számolt füzetekhez az utolsó halmaz elemeinek a számát is, megkapjuk az eredményt, ami esetünkben 10 darab füzet.

3. ábra: Összegzés halmazokkal (Forrás: Saját készítés)

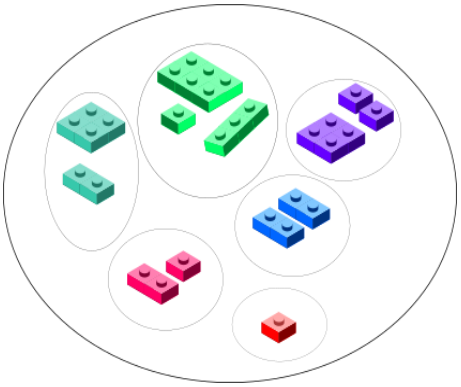
3.5.3. Minimum érték keresése

Feladat: keressük meg egy N elemű tömb minimum értékét!

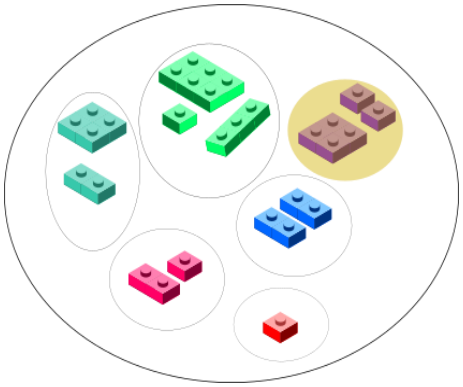
Példa: 6 különböző színt tartalmazó LEGO készletünket csoportosítottuk szín szerint. Határozzuk meg, melyik színből van a legkevesebb.

Halmazművelet: Adott N tetszőleges számú halmaz. Vizsgáljuk meg az első tetszőleges halmaz számosságát, majd hasonlítsuk össze a második halmaz számosságával. Amennyiben az első részhalmaz számossága nagyobb, mint a második elem számossága, az első elem számosságát elhagyhatjuk, és a későbbiekben a második halmaz elemszámát fogjuk hasonlítani a további választott halmazok elemszámához. Miután az aktuális legkisebb elemet összehasonlítottunk minden megmaradt halmazzal, és lecseréltük a minimumot, amikor az szükséges volt, akkor tudjuk, hogy az utolsó vizsgálatunk eredménye már a tömb minimum értéke lesz.

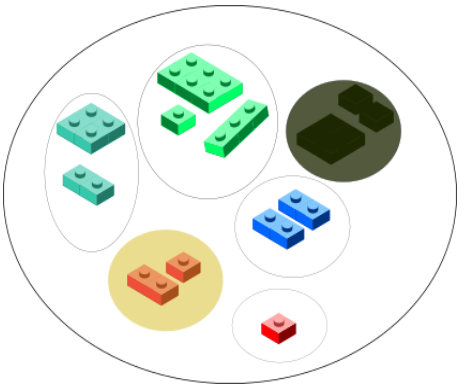
Halmazművelet magyarázata: Kezdésnek kiválasztunk egy tetszőleges számosságú részhalmazt. Miután megállapítottuk a számosságát, válasszunk ki egy másik halmazt, és vizsgáljuk meg azt is. Amennyiben az első halmaz számossága nagyobb, mint a második halmazé, akkor az utóbbit hasonlítsuk össze a harmadik elemszámával. Ha a harmadik halmaz számossága nagyobb, mint a másodiké, megállapíthatjuk, hogy a második halmaz továbbra is a minimum értéket veszi fel, ezért választhatunk egy újabbat. Miután a hatodik halmaz elemszámával is összehasonlítottuk az éppen legkisebbnek vélt halmaz számosságát, a vizsgálatunk eldönti, hogy milyen színű az a LEGO, melyből a legkevesebb van.



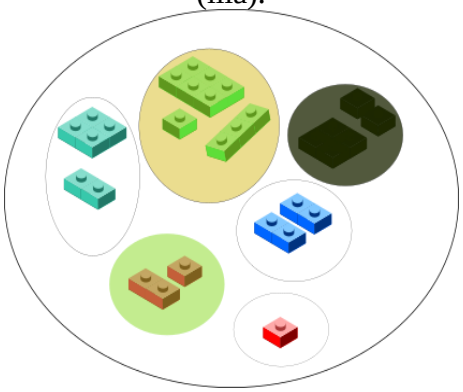
0. A LEGO halmazok definiálása.



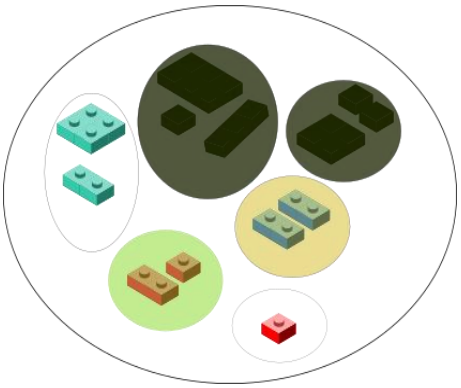
1. Kiválasztunk egy tetszőleges halmazt, és megállapítjuk az elemszámát (3) és típusát (lila).



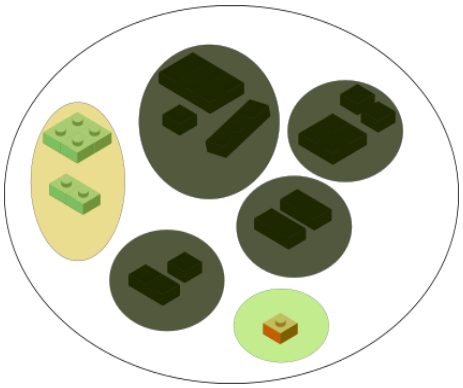
2. A megmaradt halmazokból kiválasztunk egy újabb halmazt, megállapítjuk az elemszámát és típusát (2, piros), majd összehasonlítjuk az előző választott halmazunkkal (3, lila). Mivel a pirosból kevesebb van, ezentúl azzal hasonlítjuk össze a maradék halmazokat.



3. Választunk egy újabb halmazt (3 zöld). Mivel a pirosok száma kevesebb, ezért továbbra is a piroshoz hasonlítjuk a halmazokat.



4. Kiválasztunk a megmaradt halmazokból egy újabb halmazt (kék, 2). Mivel a halmaz elemeinek száma megegyezik a pirossal, nem cseréljük ki.



5. Ha már elfogytak a halmazaink, az utolsó összehasonlítás fogja megmondani, hogy melyik halmaz tartalmazza az összes közül a legkevesebbet.

4. ábra: Minimum keresése halmazokkal (Forrás: Saját készítés)

3.5.4. Növekvő sorrendbe rendezés

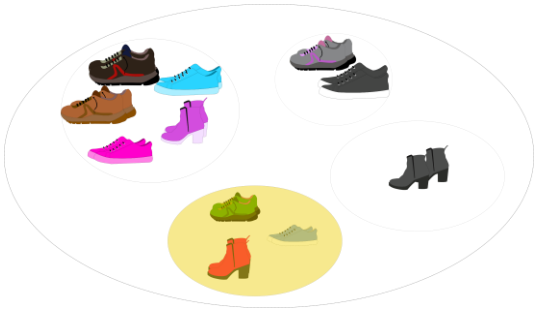
Feladat: Rendezzük egy N elemű tömb elemeit növekvő sorrendbe.

Példa: Egy háztartásban különböző méretű cipők vannak, mindegyikből több pár. Rendezzük sorba, hogy lássuk melyik méretből van legtöbb, illetve legkevesebb. (5,2,1,3)

Halmazművelet: A korábban bemutatott, minimum érték keresésére definiált eljárásunkat bővítjük. Amikor megtaláltuk azt a halmazt, ami a minimum elemszámmal rendelkezik, a maradék halmazokat egy új halmaz részhalmazaivá tesszük, és ismét kiválasztunk egy elemet közülük, és elvégezzük a minimum érték keresésére vonatkozó eljárásunkat. Ha végrehajtottuk az eljárást ismételten, már kettő elemünk van, melyek számosságát növekvő sorrendbe felírhatjuk. Ekkor alkotunk a maradék halmazoknak egy újabb összefoglaló halmazt, és ismét elvégezzük a minimum keresést. Az eredményt ismét hozzáírjuk a már rendezett számsorozatunkhoz. Addig ismételjük a korábbi műveleteket, míg végül egy halmazunk marad, amit már nem tudunk mivel összehasonlítani, ezt is hozzáírjuk az eredménylistánkhoz, és ezzel elvégeztük a sorba rendezést.

Halmazművelet magyarázata: Kezdésnek válasszunk ki egy halmazt, ami adott méretű cipők számával rendelkezik (5). A kiválasztott halmazt hasonlítsuk össze a többi halmazzal úgy, mintha a minimumot keresnénk. Ha végigértünk a halmazokon, azt a számot, ami éppen a minimumot tartalmazza, felírjuk (1), és készítünk egy újabb halmazt, amely már nem tartalmazza azt a halmazt, amelynek elemszámát felírtuk, de a maradék halmazok részhalmazai lesznek (5,2,3). Válasszunk ki egy újabb halmazt (3), majd ismét menjünk végig a halmaz elemein a minimumkeresés eljárásával. Ha végeztünk, a megtalált számot (2) írjuk az eredménylistához (1), így az már két halmaz számosságát tartalmazza (1,2). Ekkor készítünk egy újabb halmazt, ami a maradék kettő halmazt tartalmazza (5,3). Ezekből ismét válasszunk egy elemet (5), és hasonlítsuk össze a másik elemmel (3). Először a kisebb számot írjuk hozzá a listához, majd a nagyobbat, így megkapjuk a sorrendbe rendezett listát (1,2,3,5).

0. Kiinduló halmazok definiálása.



1. Válasszunk ki egy tetszőleges halmazt, és végezzük el a már ismertetett minimum kereséséről szóló eljárást.

2. Miután megtaláltuk a minimumot, vegyük ki a halmazból azt az elemet, majd végezzünk el egy minimum keresést a megmaradt 3 halmazon.

3. A megtalált minimumot ismét tegyük félre és addig hajtsunk végre újabb minimum kereséseket, míg végül már nem marad elem az eredeti halmazunkban.

4. Ha kiürült az eredeti halmazunk, akkor már csak le kell olvasnunk az eredményeket. Az először megtalált halmazunk a legkisebb-, míg az utoljára megtalált a legnagyobb elemszámú halmaz (zöld: legkisebb, piros: legnagyobb).

5. ábra: Sorba rendezés halmazokkal (Forrás: Saját készítés)

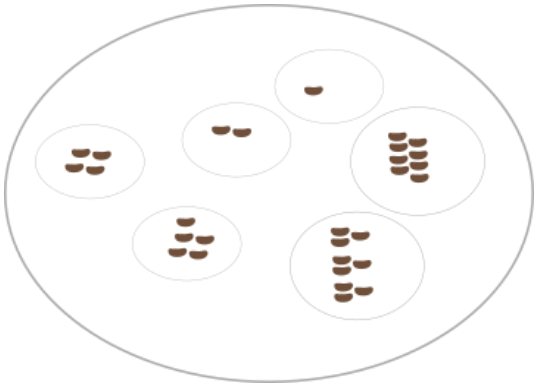
3.5.5. Komplex feladat

Az eddigi feladatokkal már szemléltettük néhány probléma megoldását (minimum keresés, sorbarendezés, osztás). Megállapíthatjuk, hogy a halmazok ezekben az esetekben jó választásnak bizonyultak, mivel a megoldási algoritmusokat fel tudtuk építeni a segítségükkel. Nézzünk meg egy komplexebb feladatot, ahol a korábban felépített megoldásainkat fel tudjuk használni.

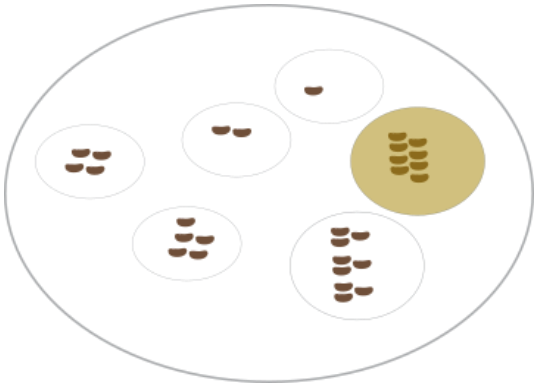
Feladat: Számoljuk ki egy N elemű tömb legkisebb és legnagyobb páros elemének szorzatát.

Halmazművelet: A korábban bemutatott sorbarendezés eljárására szükségünk lesz, de módosításra szorul. Amikor vizsgáljuk az éppen aktuális elemet, mielőtt még

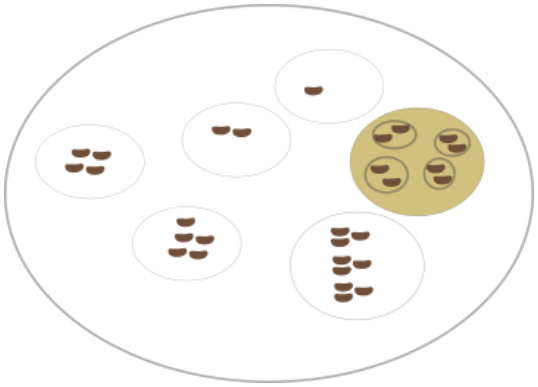
összehasonlítanánk egy másik halmazzal, alkossunk kettő elemű részhalmazokat a választott halmazon belül. A maradékos osztásból kiindulva tudjuk, hogy ha marad olyan elem a halmazban, amely nem tagja egy részhalmaznak sem, az nem páros elemű. Ha ez történt, akkor az adott halmaz kivételével alkossunk egy halmazt, melynek részhalmaza a többi halmaz, és válasszunk új halmazt. Miután így végigértünk a halmazokon, készítünk egy új üres halmazt, melybe először annyi elemet teszünk, amennyi a sorozat jobb szélső eleme annyiszor, amennyi a sorozat bal szélső számjegye. Az eredményt leolvashatjuk az újonnan keletkezett halmazról.



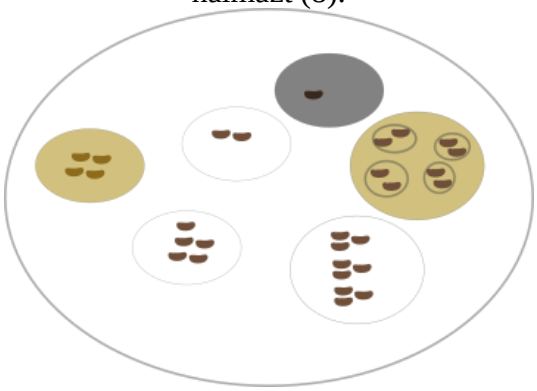
0. Definiáljuk a kiindulási állapotot.



1. Kiválasztunk egy tetszőleges halmazt (8).



2. Létrehozunk 2 elemű halmazokat, hogy a maradékos osztás vizsgálatához. Látható, hogy nincs olyan részhalmaz, mely nem azonos elemszámmal rendelkezik. Kijelenthetjük, hogy a halmaz páros elemszámú, mert maradék nélkül tudunk 2 elemű részhalmazokat definiálni.



3. Kiválasztunk egy újabb halmazt (1), és elvégezzük a 2. lépésben definiált vizsgálatot. Mivel nem tudunk 2 elemű részhalmazt létrehozni, így minimum keresésének vizsgálatát nem végezzük el, hanem választunk egy újabb halmazt (4), és a jelenlegit (1) kivesszük a halmazból.

4. Miután a halmaz minden elemén végigértünk, minimum keresésünk eredményét feljegyezzük. Majd a megmaradt elemeken folytatjuk a sorbarendezés eljárásunkat.

5. Miután végeztünk a páros elemszámú részhalmazok sorba rendezésével, létrehozzuk az eredmény halmazunkat és elhelyezünk benne 2-szer 8 elemet. A két szám a legkisebb és a legnagyobb elemszámú halmazunk számossága.

6. ábra: Komplex feladat megoldása halmazokkal (Forrás: Saját készítés)

Úgy gondolom, hogy feltételezésem igazolást nyert, mely szerint a halmazokat tudtam egyfajta absztrakt platformként használni, hogy a feladatban lévő információkat jobban megfigyelhessük, majd feldolgozhassuk. Így a problémák megoldása is könnyebbé vált. A feladatokat megoldottuk a bemutatott módszerekkel, de fontos megjegyezni, hogy a szimbolika és a halmaz reprezentáció bár megkönnyítette a feladatmegoldásokat, egy fontos lépés még hátra van: a programozási nyelvben való implementáció.

Az implementációt segíti, ha felismerjük, hogy a halmazok rendezésekor pontosan milyen műveleteket hajtunk végre, és azok a programozásban milyen szimbólumoknak felelnek meg. Anélkül, hogy elmélyülnénk egy tetszőleges programozási nyelv szintaktikájában, láthatjuk, hogy egy halmaz megfelel egy tömbnek. Ugyancsak megállapítható, hogy az implementáció során az ismételt tevékenységek az úgynevezett ciklusoknak feleltethető meg.

Választott absztrakt platformunk azzal a további előnnyel rendelkezik, hogy tulajdonképpen a számokat magukat reprezentálhatjuk halmazok formájában. Változik-e a feladatunk, ha a számok nem kézzel fogható tárgyakat, hanem mértékegységeket jelölnek? Van-e különbség, ha a feladat nem valóságban érzékelhető eszközöket, hanem fizikai egységeket tartalmaz?

A válasz kézenfekvő, ha figyelembe vesszük, hogy a halmazok számosságát nem az határozza meg, hogy mit tartalmaznak, hanem az, hogy mennyit (Skemp 2005: 241). Ebből az következik, hogy különböző magasságú emberek magassága vagy LEGO-k darabszáma szerinti sorba rendezése az alapján, hogy melyikből mennyi van, ezen az absztrakciós szinten nem lényeges. Azért választottam a LEGO-kat, mert azt feltételezem, hogy azokat egyszerűbb elképzelni, mint a mértékegységek egységeit.

Azt gondolom, hogy Dienes professzor játékai (Dienes 2018) is az előbb kifejtett gondolatra építenek, miszerint az egységek nem relevánsak, ezért azt a tapasztalatot kell átadni, amelyből az absztrakció kikövetkeztethető.

4. Informatika érettségi feladatok megoldása halmazokkal

Ebben a fejezetben korábbi informatika érettségi feladatok közül válogatok, melyeket halmazok segítségével értelmezek. Látni fogjuk, hogy a korábban bemutatott módszer itt is segítségünkre lehet.

4.1. Érettségi feladat I.

A feladatban az első osztályú labdarúgó-bajnokság csapatai számára kell adatbázist létrehoznunk, és ehhez lekérdezéseket írnunk. Az adatbázis létrehozásának műveletére nem térnék ki, mert az nem képezi részét a megoldandó logikai problémáknak. (2018tk: 10)

Feladat: Vizsgáljuk meg a feladatot és készítsük el halmaz reprezentációját.
Feladat értelmezése: Három különböző típusú halmazunk van (labdarúgó, klub, poszt). A klub részhalmazai a csapatok, melyek számosságát a csapatba tartozó labdarúgók száma határozza meg. Rendelkezünk egy poszt halmazzal, amelynek részhalmazai a posztok megnevezései, melyek szintén a labdarúgókat tartalmazzák. Láthatjuk, hogy minden labdarúgó tartozik egy poszt halmazhoz, valamint egy klub halmazhoz is[3]. Észrevehetjük továbbá, hogy minden játékosnak pontosan egy posztja és egy csapata van.
A csapat és a poszt halmazoknak a részhalmazait labdarúgók *partíció*jának nevezzük. Az említett halmazok részhalmazait (pl.: kapus poszt) a labdarúgók *ekvivalencia osztályainak* tekintjük. Ha megvizsgáljuk a labdarúgók táblát, láthatjuk, hogy több tulajdonsága is van 1-1 játékosnak. Célszerű, hogy a labdarúgók halmazába csak a játékosok egyértelmű azonosítói legyenek benne, és a tulajdonságaik lehetséges értékei számára alkossunk külön halmazokat, és a rendszer könnyebben áttekinthető marad. Így létrehozzuk a mezszám, utónév, vezetéknév, születési idő, magyar, külföldi, valamint érték halmazokat, amelyek tartalmaznak minden

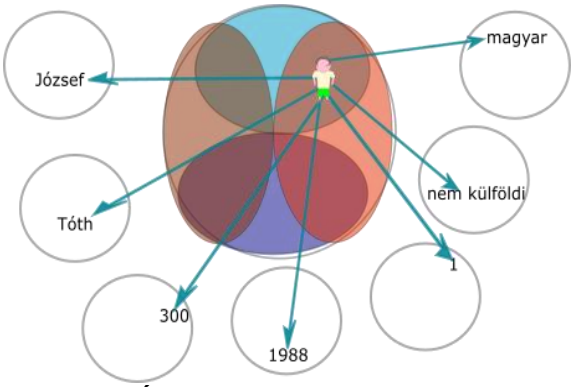
3 A megállapításban felismerhetjük az elsődleges, valamint az idegen kulcs fogalmát. Ha lehet olyan játékosunk, amely nem tagja egyetlen csapatnak sem, akkor nincs idegen kulcsként definiálva a csapat azonosítója a labdarúgók esetében.

előforduló értéket, amelyek a játékosok tulajdonságai. Ezeket a halmazokat a játékosok *leképezései*nek nevezzük.

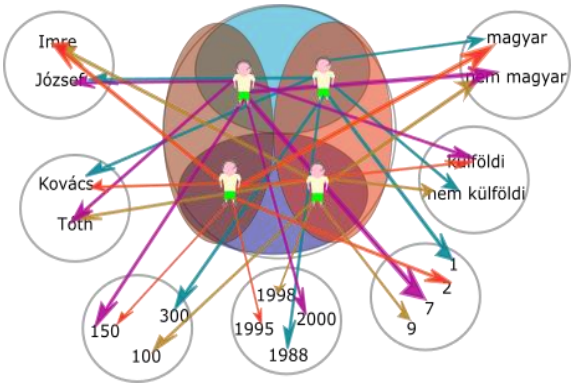
Az adott játékos és leképzése (pl.: a játékos értéke) közötti kapcsolatot nyíllal jelöljük, amely a játékosból mutat az értékére.

Az előbbiekben felvázolt folyamatot a matematikában úgy nevezik, hogy az „univerzum meghatározása” (Skemp 2005:195).

Mivel az adatbázis és adattáblák fogalmait nem értelmezzük ezen az absztrakciós szinten, ezért halmazokkal reprezentáljuk az egyes értékeket. Később ezeket az egységeket adatbázis típusától[4] függően leképezhetjük. Elemezve az elkészült ábráinkat láthatjuk, hogy az adatok korlátozottan olvashatóak nagy elemszám esetén, de azok megértéséhez szerencsére nem szükséges sok adat.



1. Ábrázoljuk a feladatot egy darab elemmel. A kék halmazok a csapatokat, a piros árnyalatúak a posztokat jelölik. A nyilak a játékos tulajdonságainak értékeire mutatnak.



2. Helyezzünk el további elemeket a rendszerünkben. A 4 játékosunk értékeivel bővítettük leképezés halmazokat. Minden játékosról meg tudjuk állapítani, hogy melyik csapatban, illetve poszton játszik, ha megnézzük melyik részhalmazok eleme.

7. ábra: Focisták definiálása halmazokban (Forrás: Saját készítés)

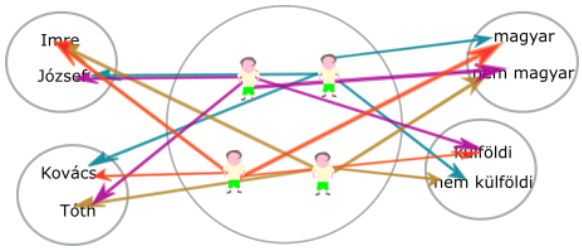
Feladat: „Készítsen lekérdezést, amely megjeleníti azon játékosok vezeték- és utónevét, akiknek magyar és külföldi állampolgárságuk is van!”

Feladat értelmezése: Látjuk, hogy a leképezések korábbi felrajzolása módosításra szorul a labdarúgó magyar és külföldi tulajdonságának esetében. A hibát könnyen orvosolhatjuk, ha a labdarúgókat két részhalmazra bontjuk (magyar és külföldi)[5].

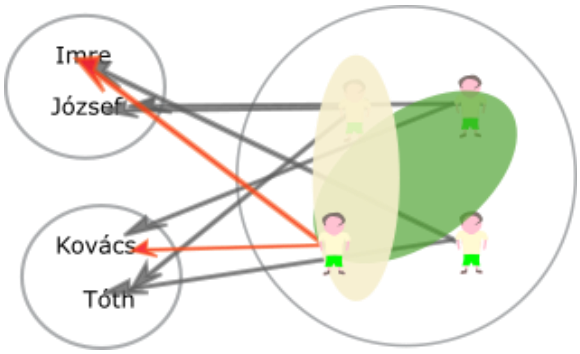
4 Jelen esetben egyszerűen csak NoSQL vagy SQL-re gondolok.

5 Felismerhetjük a normalizálás folyamatát is ebben a műveletben.

Miután felbontottuk a labdarúgókat aszerint, hogy ki tartozik a magyar és ki a külföldi játékosokhoz, a két halmaz metszetében lévő elemek megfelelnek a feladat megoldásának. Ezen elemek leképezéseit (vezeték- és utónév) az elemekből kiinduló nyilak követésével tudjuk leolvasni az ábránkról.



1. Rendezzük az ábránkat úgy, hogy csak a szükséges adatok látszódjanak. (külföldi, vezetéknév, magyar, keresztnév)

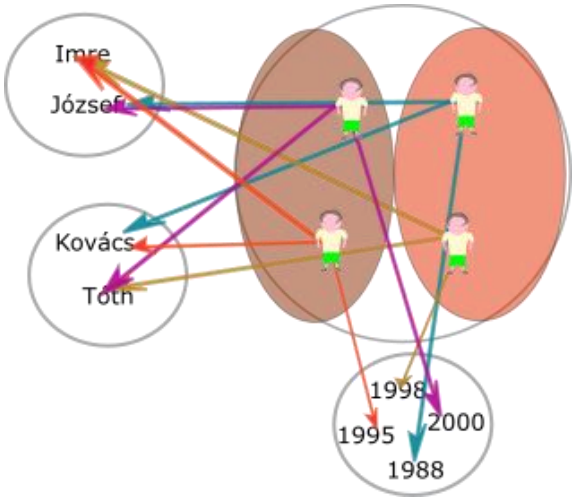


2. Hozzunk létre részhalmazokat a játékosok magyar és külföldi leképezéseik alapján. A két halmaz metszetében találjuk az eredményeket.

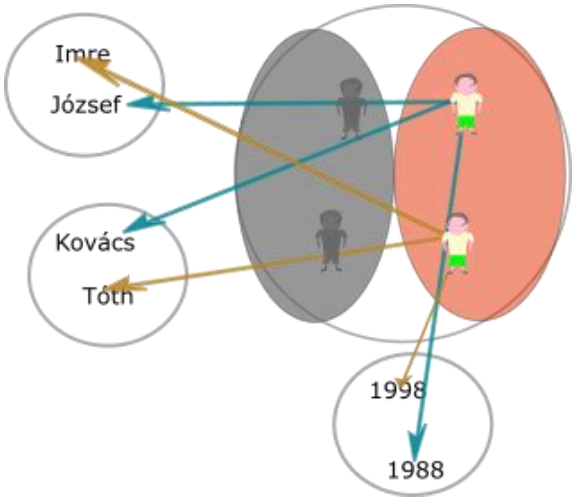
8. ábra: Külföldi és magyar focisták szűrése halmazokkal (Forrás: Saját készítés)

Feladat: „A kapusokon kívül mindenkit mezőnyjátékosnak tekintünk. Készítsen lekérdezést, amely megadja a legidősebb mezőnyjátékos vezeték- és utónevét, valamint születési dátumát! (Feltételezheti, hogy csak egy ilyen játékos van.)”

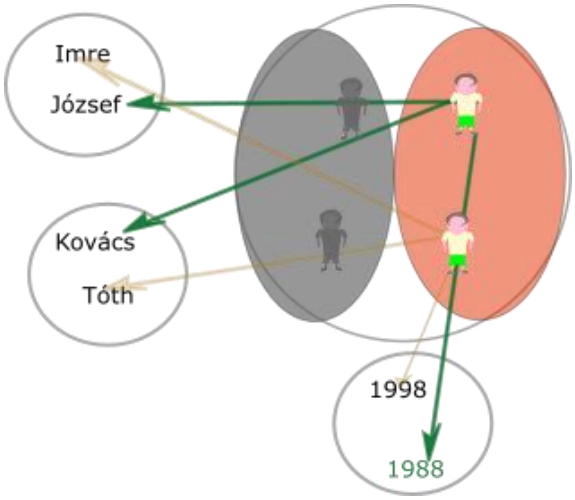
Feladat értelmezése: A labdarúgók halmazából vegyük ki azokat az elemeket, amelyek a poszt kapus halmazának elemei. Ez alapján frissítsük a leképezéseket is a (kiemelten fontos a születési idő), hogy a lehetséges születési idők halmaza ne tartalmazza a kivett labdarúgó(k)hoz tartozó értéke(ke)t. A korábbi fejezetben tárgyalt módon keressük meg a születési idő halmazában azt az értéket, amely a legkisebb (legkorábbi dátum). A megtalált elem a feladat leírása szerint pontosan 1 elemnek feleltethető meg a labdarúgók halmazában. Miután visszakövettük, hogy mely játékoshoz tartozik a legkorábbi születési idő, megnézzük azt, hogy a megtalált játékoshoz milyen vezetéknév és utónév tartozik.



1. Rendezzük az ábránkat, hogy csak a szükséges adatok látszódjanak. (születési idő, vezetéknév, keresztnév, poszt részhalmazok)



2. Távolítsuk el a kapu poszthoz tartozó játékosokat a hozzájuk tartozó leképezésekkel.

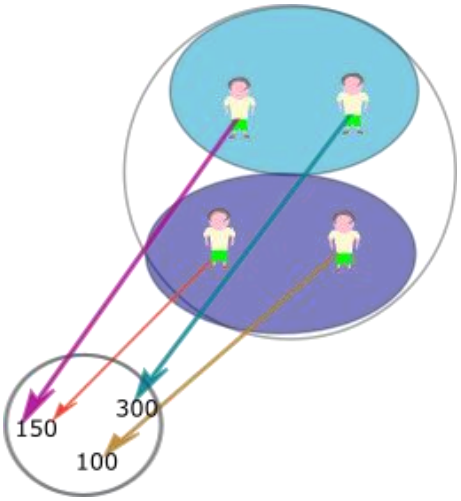


3. Keressük meg a születési idő halmaz minimumát a már ismertetett módon, és olvassuk le az évszámhoz tartozó játékos adatait.

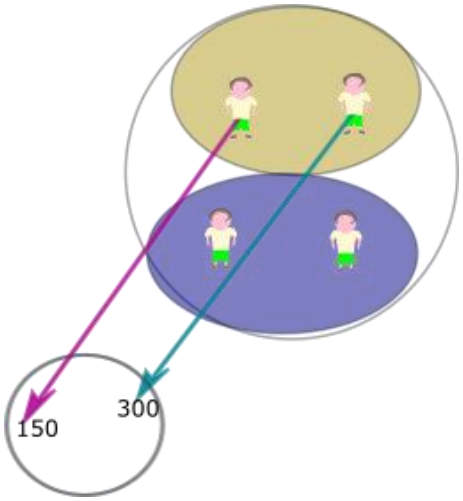
9. ábra: A legidősebb focista adatai (Forrás: Saját készítés)

Feladat: „Készítsen lekérdezést, amely csapatonként megjeleníti a játékosok összértékét! A csapatok neve és a játékosainak összértéke jelenjen meg!”

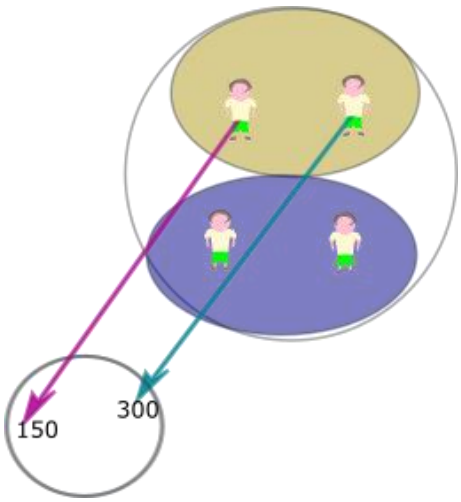
Feladat értelmezése: A feladatban frissítjük a labdarúgók halmazát aszerint, hogy mindig csak egy csapat játékosai látszódjanak, majd a leképezések frissítése után összegezzük az értékek halmazát az előző fejezetben tárgyalt módon. A részeredmények összegyűjtése után megkapjuk a feladat megoldását.



1. Rendezzük az ábránkat, hogy csak a szükséges adatok látszódjanak. (játékosok, csapatok, értékek)



2. Válasszuk ki egy tetszőleges részhalmazát a játékosok halmazának, és frissítsük a megmaradtak szerint a leképezést.

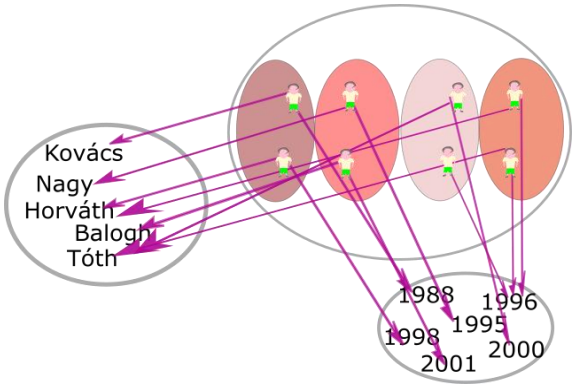


3. Összegezzük az érték halmaz elemeit. A továbbiakban addig ismételjük a kettes és a hármas lépést, amíg végig nem értünk minden csapat részhalmazán a játékosoknak.

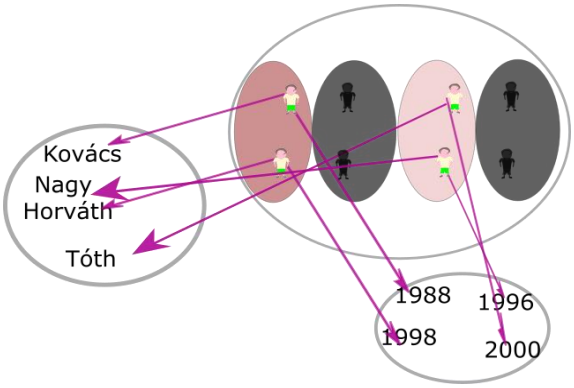
10. ábra: A játékosok értékei csapatok alapján (Forrás: Saját készítés)

Feladat: „Készítsen lekérdezést egy külföldi menedzser számára, amely megad minden olyan kapust és bal oldali játékost, aki 1998-ban vagy később született! A lekérdezésben jelenjen meg a vezetéknév, a poszt és a születési idő!”

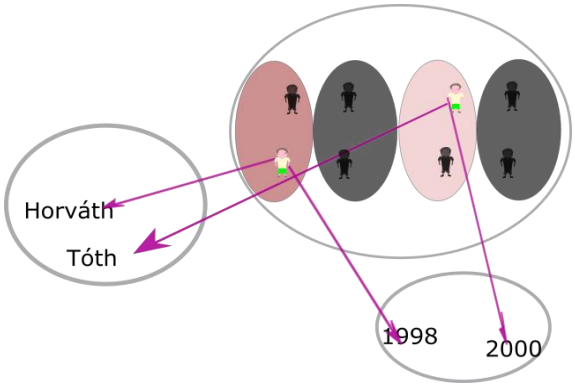
Feladat értelmezése: Első lépésként frissítsük a labdarúgók halmazát és a *leképezéseit* az alapján, hogy akik nem részei a poszt *partíció* bal oldali, vagy a kapu *ekvivalencia osztály*ának, azokat kivesszük. A labdarúgók születési idejének leképezését vizsgáljuk és kivesszük a 1998 előtti dátumokat és a hozzájuk tartozó labdarúgókat. A megmaradt labdarúgókat vizsgáljuk, hogy a poszt melyik részhalmazában vannak, valamint a leképezéseik közül vezetékneveiket és születési idejeiket.



1. Rendezzük az ábránkat, hogy csak a szükséges adatok látszódjanak. (születési idő, posztok, vezetéknév)



2. Vegyük ki azokat a poszt részhalmazokat, melyek nem baloldali, vagy kapus játékosokat tartalmaznak

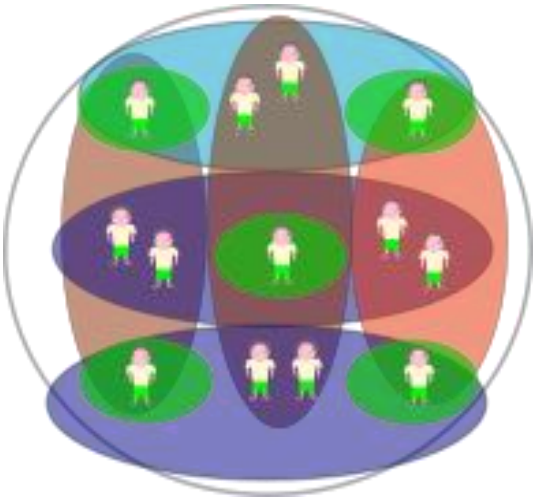


3. Válogassuk ki a születési idők közül az 1998 utániakat, és vegyük ki a hozzájuk tartozó labdarugókat. A megmaradt játékosok képezik az eredményhalmazt.

11. ábra: A játékosok értékei csapatok alapján (Forrás: Saját készítés)

Feladat: „A klubok a játékoskeretük kialakítása során arra törekednek, hogy az egyes posztokra legalább két játékost szerződtessenek. Készítsen lekérdezést, amely megadja, hogy mely csapatoknál mely posztokon van csupán egy szerződtetett játékos!”

Feladat értelmezése: Vizsgáljuk meg a labdarúgók azon részhalmazait, melyek számossága pontosan 1. Mivel a csapat *ekvivalencia osztályainak* labdarúgó elemeinek mindegyike tartozik egy poszt partíció részhalmazához is, így az átfedéseket könnyedén megtaláljuk.

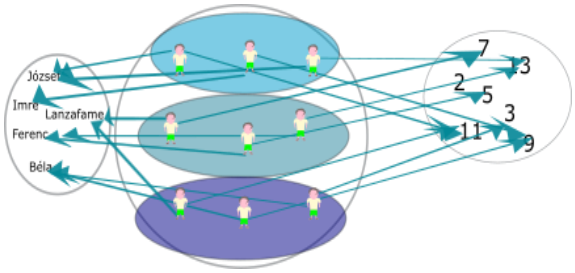


Miután rendeztük az ábránkat, vizsgáljuk meg a poszt részhalmazok és a csapatok metszeteit. A keresett halmazok, azok a metszetek, ahol csak 1 játékos található (zölddel jelölt).

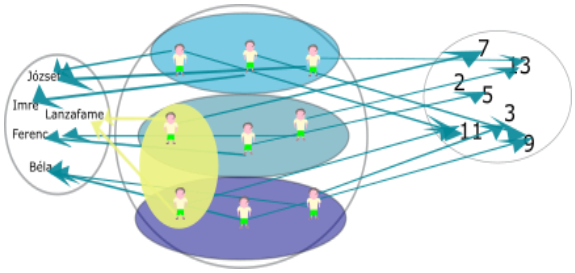
12. ábra: Az egy játékost tartozó posztok és csapatok (Forrás: Saját készítés)

Feladat: „Készítsen lekérdezést, amely felsorolja azon klub játékosait, ahol a Lanzafame vezetéknevű játékos is szerepel! A mezszámot és a vezetéknevet jelenítse meg!”

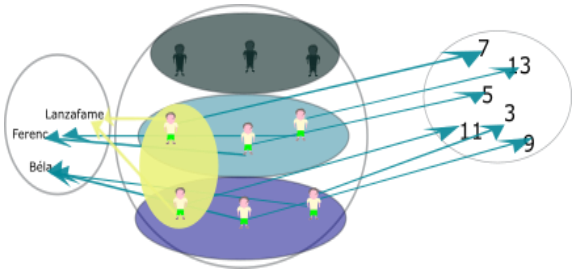
Feladat értelmezése: Vizsgáljuk meg, hogy a labdarúgók vezetéknév leképezései közül mely elemek tartoznak „Lanzafame”-hez. Ezután a labdarúgókba alkossunk egy újabb részhalmazt, mely ezeket az elemeket tartalmazza. Keressük meg azokat csapatokat, amelyekkel van közös metszete a most létrehozott halmazunknak és nem üres halmaz. Az így megtalált elemeknek olvassuk le a mezszám, illetve a vezetéknév leképezéseit a már ismert módon.



1. Rendezzük az ábránkat, hogy csak a szükséges adatok látszódjanak. (vezetéknév, csapat, mezszám)



2. Keressük meg a Lanzafame játkosokat, és alkossunk velük egy halmazt.



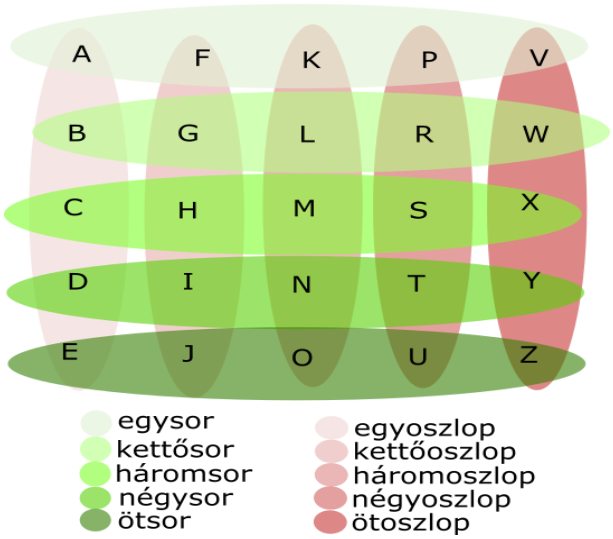
3. Vegyük ki azokat a halmazokat, amelyeknek nincs metszete az új halmazunkkal. A megmaradt játékosokhoz tartozó adatok lesz a feladatunk megoldása.

13. ábra: Lanzafame játékosok csapatainak a játékosai (Forrás: Saját készítés)

4.2. Érettségi feladat II.

Playfair-négyzet kódolási módszer implementálása a feladat. A nyelvi absztrakcióhoz tartozó feladatok értelmezésére nem térek ki. (2020te: 6)

Feladat: Vizsgáljuk meg a feladatot és készítsük el halmaz reprezentációját.
Feladat értelmezése: Kezdetnek alkossunk egy halmazt, amely 25 karaktert tartalmaz, és az angol ábécé betűiből áll, de nem tartalmazza a Q karaktert. Mivel egy mátrix megalkotása a cél, a halmaznak definiálunk 2 darab ekvivalencia osztályt, melyek a sorok és az oszlopok. Egy karakter kiválasztásakor meg tudjuk határozni, hogy mely ekvivalencia-osztályba tartozik. Annak érdekében, hogy a halmazok elemei és megnevezései ne keveredjenek, célszerű számok neveivel jelölni a halmazokat („egy", „kettő", „három", „négy", „öt"), amelyeket kiegészítünk a partíció elnevezésével („egyoszlop", „kettősor" stb.). A megnevezések azért fontosak, hogy még véletlenül se gondoljunk a halmazok számosságára. A partíciók létrehozása után mind a 25 elem tagja két részhalmaznak.

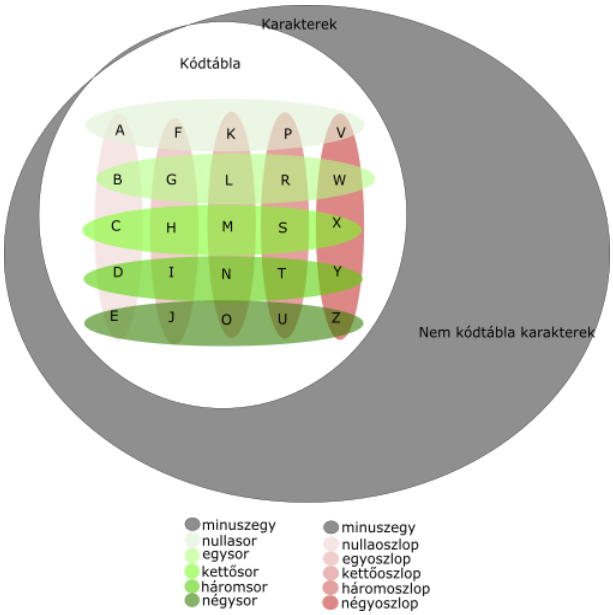


14. ábra: Kódtábla halmaz reprezentációja (Forrás: Saját készítés)

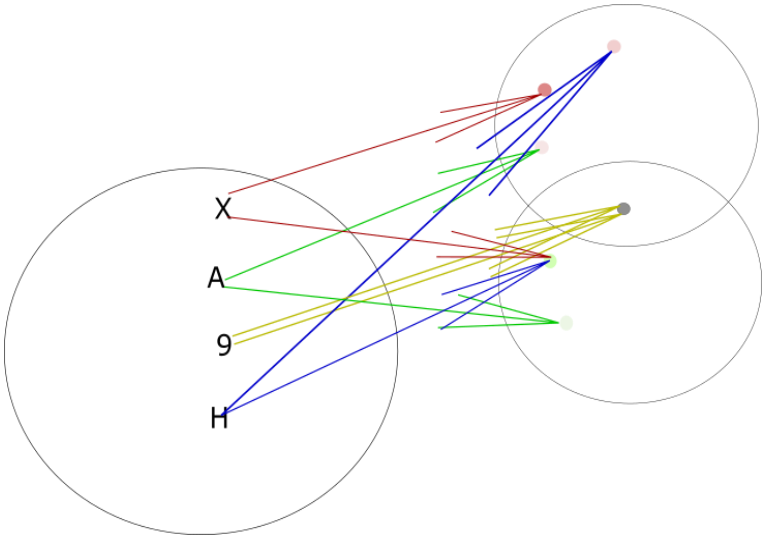
Feladat: „Készítsen a PlayfairKodolo osztályba egész típusú adattal visszatérő metódusokat SorIndex() és OszlopIndex() azonosítóval, melyek a paraméterben megadott nagybetűs karakter helyét (indexét) határozzák meg értelemszerűen a kulcstáblában! A sorok és oszlopok

indexelése nullával indul. Ha a megadott karakter nem található a kulcstáblában, akkor a metódusok -1 értékkel térjenek vissza!”

Feladat értelmezése: Korábban megalkotott halmazábránkat javítanunk kell. A kulcstábla halmazunk egy nagyobb halmaz részhalmaza, mely több karaktert is tartalmaz. Adjunk hát hozzá egy új halmazt, amely ekvivalens a következő halmazokkal: „minuszegysor”, „minuszegyoszlop”, “minuszegy”. Létrehozott halmazunk szintén a sor, illetve az oszlop ekvivalenciaosztályai közé tartoznak. Módosítsuk a partícióinkat aszerint, hogy ne „egy”-től induljanak, hanem „nulla”-tól. A feladatban meghatározott „SorIndex() metódus” megfelel annak a tevékenységnek, amikor kiválasztunk egy betűt, és megvizsgáljuk, hogy mely ekvivalenciaosztálynak tagja a sor partícióban. Előbbi megállapításunk igaz az „OszlopIndex() metódus”-ra is, azzal a különbséggel, hogy az oszlop partíciót vizsgáljuk.



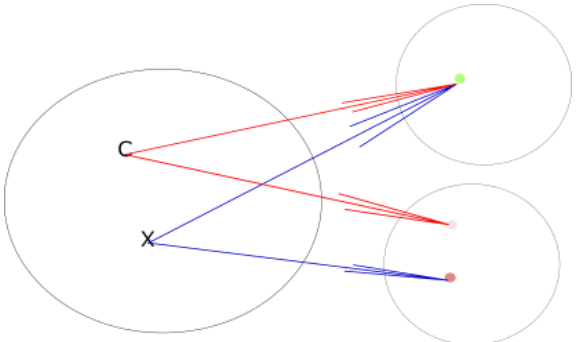
15.1. ábra: Karakterek halmazának bővítése (Forrás: Saját készítés)



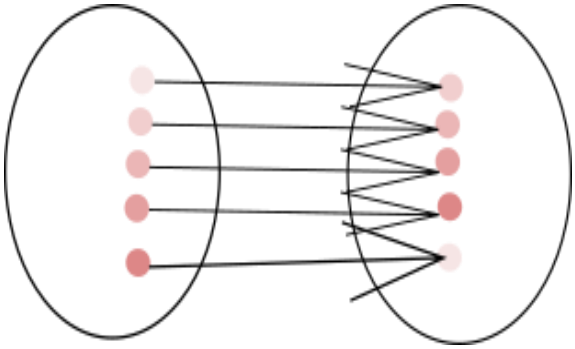
15.2. ábra: Karakter ekvivalencia-osztályainak meghatározása (Forrás: Saját készítés)

Feladat: „Ha a betűpár mindkét betűje ugyanabban a sorban jelenik meg a kulcstáblán, akkor a tőlük közvetlenül jobbra állóval kell helyettesíteni őket! … Ha történetesen az egyik betű a sor jobb szélén van, akkor a sor bal szélén álló betűvel kell helyettesíteni!”

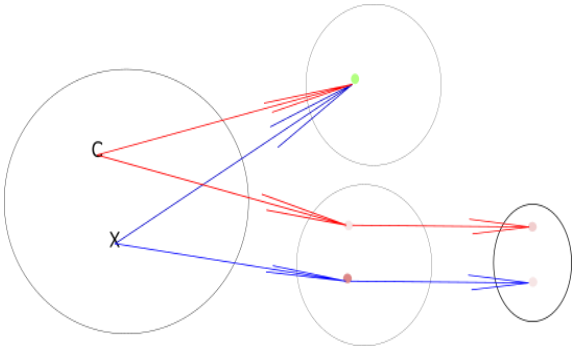
Feladat értelmezése: Válasszunk ki két karaktert, ha ezek azonos sor ekvivalencia-osztályban vannak, akkor vizsgáljuk meg melyik részhalmazában találhatóak az oszlop partíciónak. Ha nem a „négyoszlop”, akkor a várt karakter egy szomszédos ekvivalencia-osztályban található, melynek a neve megegyezik az egyel nagyobb szám nevével jelzett részhalmaz nevével („egyoszlop” → „kettőoszlop”). Ha a választott karakterünk a „négyoszlop” *ekvivalencia-osztály*ban található, akkor a keresett karakter az, amely a „nullaoszlop” és az azonos sor részhalmazában található.



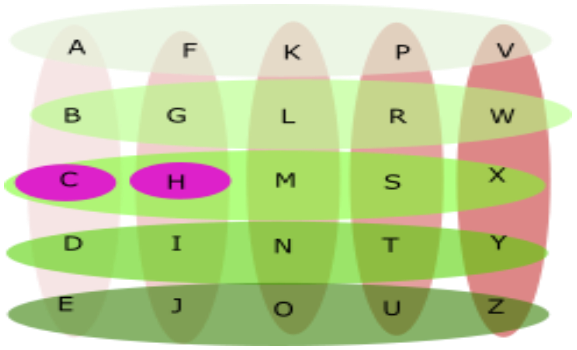
1. Határozzuk meg a karaktereket. Vizsgáljuk meg, hogy azonos ekvivalencia-osztályban vannak-e.



2. Definiáljuk a szabályt.



3. Egészítsük ki a szabállyal a kiindulási ábránkat.

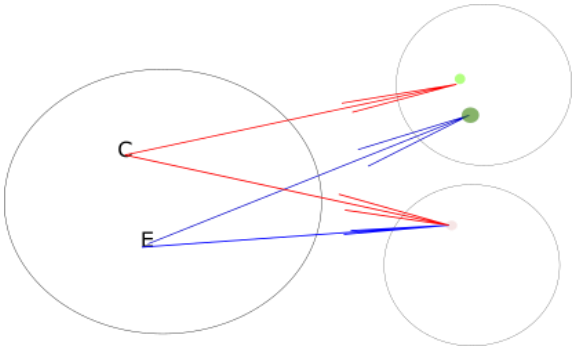


4. Jelöljük meg a kimeneti karaktereket a kódtáblában (lila).

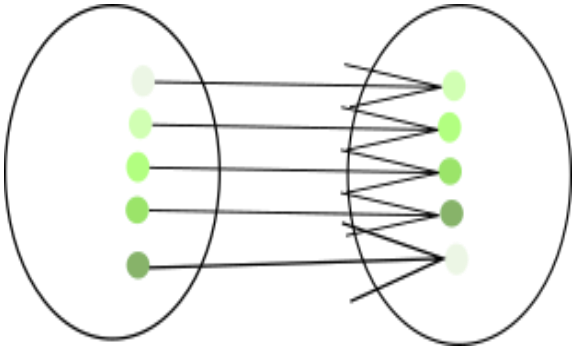
15. ábra: Kimeneti karakter meghatározása azonos sor esetén (Forrás: Saját készítés)

Feladat: „Ha a betűpár mindkét betűje ugyanabban az oszlopban jelenik meg a kulcstáblán, akkor közvetlenül az alattuk állóval kell helyettesíteni őket! … Ha az egyik betű az oszlop alján van, akkor az oszlop tetején álló betűvel kell helyettesíteni!”

Feladat értelmezése: Feladatunk megegyezik az előzővel annyi különbséggel, hogy először nem sort, hanem oszlopot vizsgálunk és úgy határozzuk meg a karakter pozícióját. Amennyiben a „négysor” karakteréről beszélünk, akkor viszont a „nullasor” -t vesszük.[6]

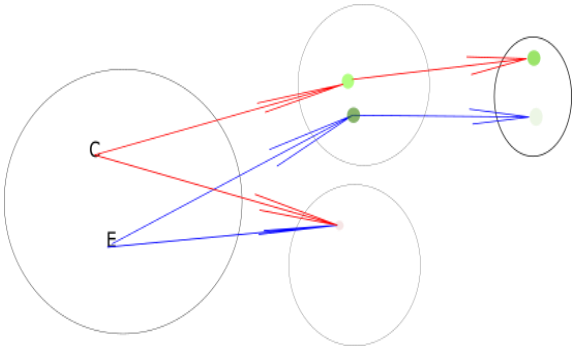


1. Határozzuk meg a karaktereket. Vizsgáljuk meg, hogy azonos ekvivalencia-osztályban vannak-e.

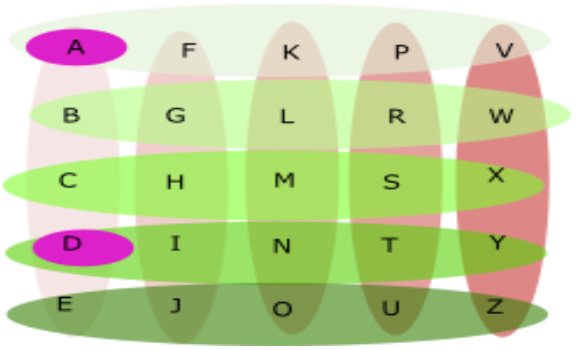


2. Definiáljuk a szabályt.

6 Programozásban ez egy jelzés, hogy vélhetően közös eljárást alkalmazhatunk a két tevékenységre.



3. Egészítsük ki a szabállyal a kiindulási ábránkat.

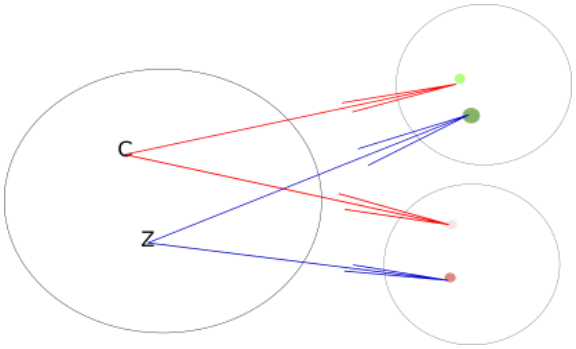


4. Jelöljük meg a kimeneti karaktereket a kódtáblában (lila).

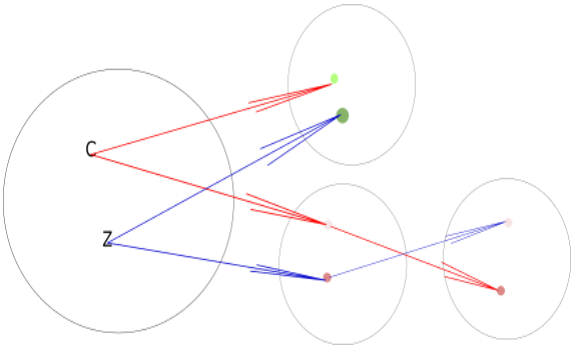
16. ábra: Kimeneti karakter meghatározása azonos oszlop esetén (Forrás: Saját készítés)

Feladat: „Ha a betűpár betűi nincsenek sem egy sorban, sem egy oszlopban, akkor tekintsük azt a kulcstábla mezőiből felépülő téglalapot, amelynek a két betű a két szemközti csúcsa. A betűket a saját sorukban, a téglalap másik csúcsánál lévő betűkkel helyettesítjük.”

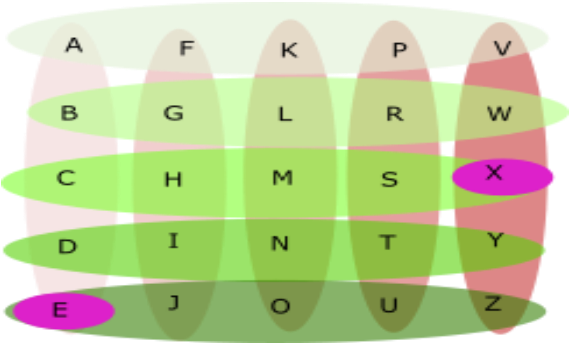
Feladat értelmezése: Válasszunk két karaktert, ha ezek különböző sor- és oszlop ekvivalencia-osztályban vannak, akkor úgy kapjuk meg a várt karaktereket, hogy miután megvizsgáltuk a két karakter helyzetét, felcseréljük oszlop *ekvivalencia-osztályaikat*.



1. Határozzuk meg a karaktereket.



2. Cseréljük meg az oszlop ekvivalencia-osztályokat.



4. Jelöljük meg a kimeneti karaktereket a kódtáblában (lila).

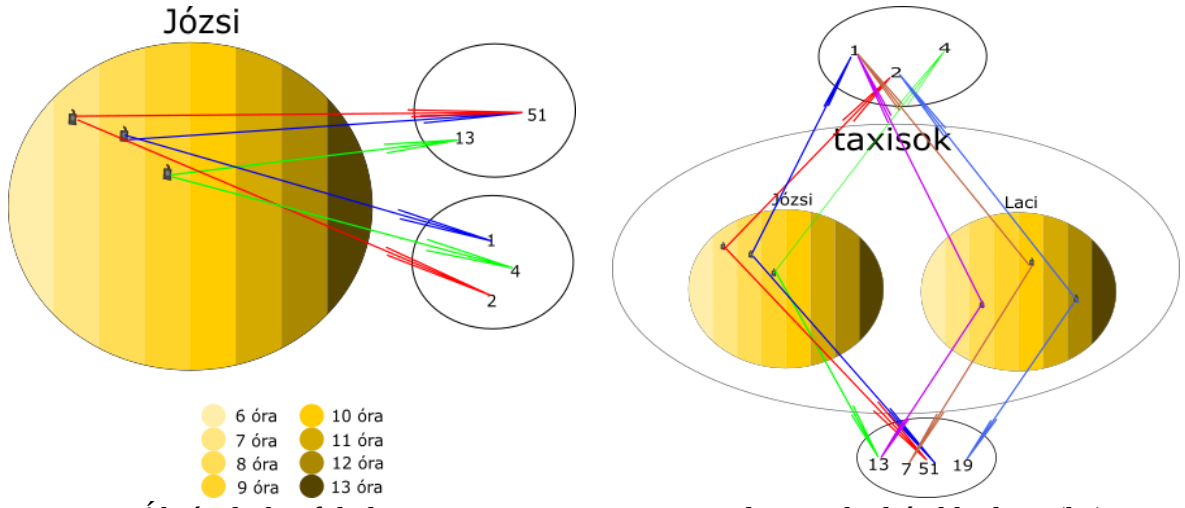
17. ábra: Kimeneti karakter meghatározása különböző sor és oszlop esetén (Forrás: Saját készítés)

4.3. Érettségi feladat III.

A feladat egy CB-rádió naplózási adatainak feldolgozása. Bár a feladat nem adatbázis kezeléssel kapcsolatos, megfigyelhetjük, hogy ezen az absztrakciós szinten közel azonos gondolkodási műveletek jellemzik. (2019ok: 6)

Feladat: Vizsgáljuk meg a feladatot és készítsük el halmaz reprezentációját.

Feladat értelmezése: Első lépésként ábrázoljuk a taxisofőrök halmazait, amelyeknek neve a taxisofőr beceneve. A halmazokon hozzunk létre partíciót a bejegyzések órája alapján (6,7,8,9,10,11,12,13). Mivel a percek értéke 0-59 közötti értéket vehet fel, ezért nem célravezető, hogy a perceknek is külön *ekvivalencia-osztályokat* definiáljunk (60-at). A taxisofőrök elemei az egy percen belül indított hívások. Ezeket leképezhetjük a percek értékével, hiszen az adott percen belül indított hívás magába foglalja azt a percet is, amikor indítottuk. Ugyancsak leképezése a hívásoknak az egy percen belül indított hívások száma.



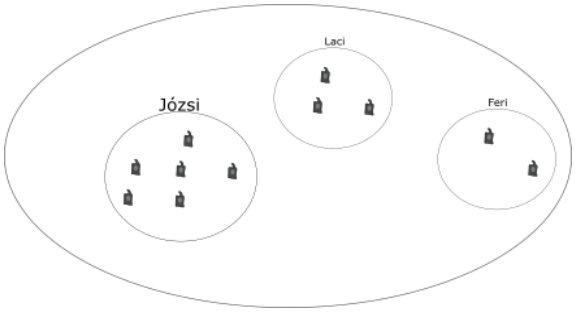
1. Ábrázoljuk a feladatot egy darab elemmel.

2. Helyezzünk el újabb eleme(ke)t a rendszerünkben.

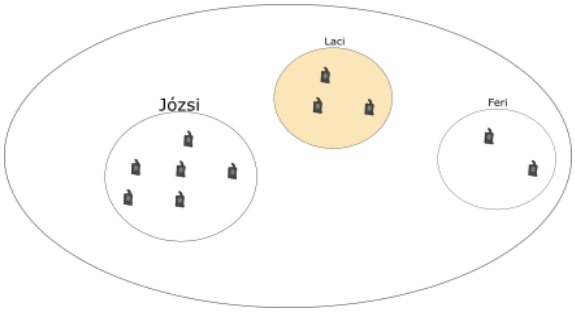
18. ábra: Taxisofőrök halmaz reprezentációja (Forrás: Saját készítés)

Feladat: „Határozza meg és írja ki a képernyőre a minta szerint, hogy hány bejegyzés található a forrásállományban!”

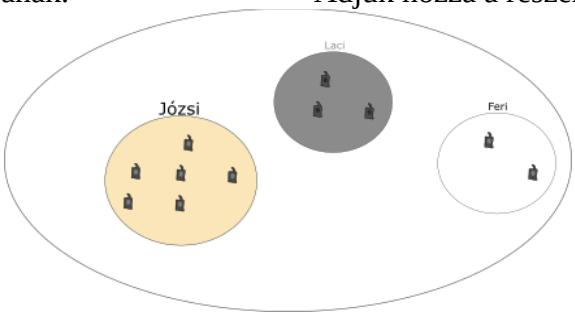
Feladat értelmezése: Mint azt korábban már tárgyaltuk egy hasonló feladat kapcsán, egyszerű dolgunk van. Egyesével válasszuk ki a taxisofőrök halmazát és elemeik számát összegezzük.



1. Rendezzük az ábránkat, hogy csak a szükséges adatok látszódjanak.



2. Tetszőleges halmaz kiválasztása után, számoljuk meg az elemeinek a számát (3). Adjuk hozzá a részeredményünkhöz (0+3).

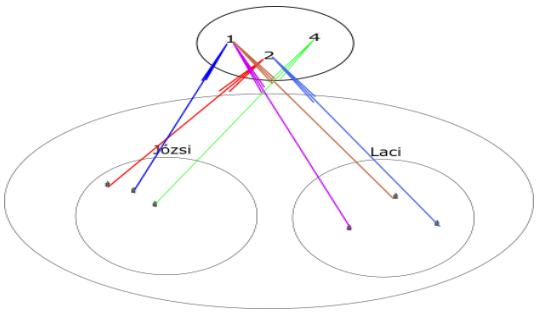


3. A megmaradt halmazokból ismét válasszunk ki egy halmazt (6), és adjuk hozzá a korábbi részeredményünkhöz (3+6). Addig ismételjük ezt a lépést, míg marad olyan halmaz, amely elemeinek a számát, még nem adtuk hozzá a részeredményünkhöz.

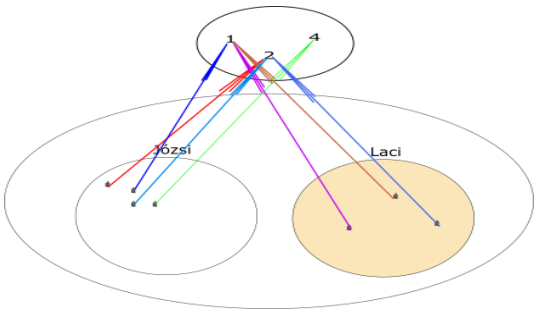
19. ábra: Taxisofőrök bejegyzéseinek megszámlálása (Forrás: Saját készítés)

Feladat: „Döntse el és írja ki a képernyőre a minta szerint, hogy található-e a naplóban olyan bejegyzés, amely szerint a sofőr egy percen belül pontosan 4 adást indított! A keresést ne folytassa, ha az eredményt meg tudja határozni!”

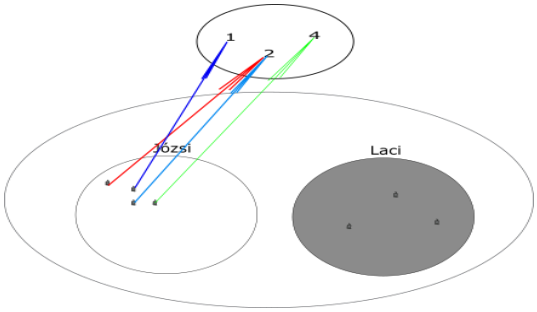
Feladat értelmezése: Vizsgáljuk a taxisofőrök elemeit egyesével. Amennyiben találunk egy 4-es számot, megtaláltuk a keresett taxisofőrt.



1. Rendezzük az ábránkat, hogy csak a szükséges adatok látszódjanak. (taxisofőrök, hívások száma percenként)



2. Válasszunk ki egy taxisofőrt. Vizsgáljuk meg a hívásaihoz tartozó hívás szám leképezéseket.

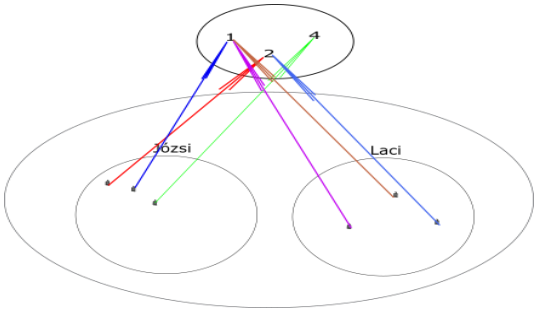


3. Amennyiben az előző lépésben nem találtunk olyan hívást, amelyhez 4 hívás tartozik, válasszunk egy újabb halmazt vizsgálatra. Ezt a lépést addig ismételjük, míg el nem fogynak a halmazok, vagy nem találunk egy olyan elemet, amelyhez 4 hívás tartozik.

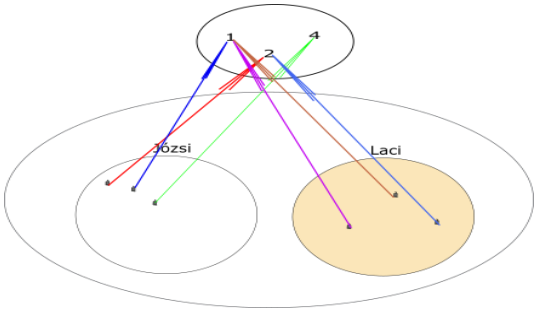
20. ábra: 4 hívással rendelkező taxisofőr keresése (Forrás: Saját készítés)

Feladat: „Kérje be a felhasználótól egy sofőr nevét, majd határozza meg a sofőr által indított hívások számát a napló bejegyzéseiből! Az eredményt a minta szerint írja ki a képernyőre! Ha olyan sofőr nevét adja meg a felhasználó, aki nem szerepel a naplóban, akkor a „Nincs ilyen nevű sofőr!” mondat jelenjen meg!”

Feladat értelmezése: Keressük meg azt a halmazt, amelynek neve megegyezik a bekért névvel. Amennyiben megtaláltuk, összesítsük az általa indított hívásokat a korábban már ismertetett módon.



1. Rendezzük az ábránkat, hogy csak a szükséges adatok látszódjanak.

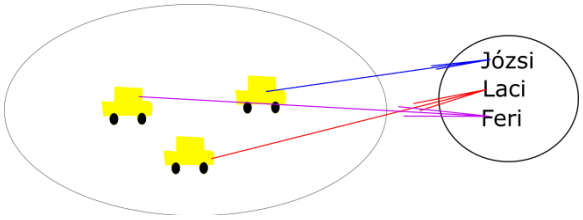


2. Válasszunk ki egy halmazt. Amennyiben a taxisofőr neve megegyezik a bekért névvel, összegezzük a hívásaihoz tartozó értékeket. Ha a neve nem egyezik a bekért névvel, válasszunk egy újabb halmazt.

21. ábra: Taxisofőr keresése név szerint (Forrás: Saját készítés)

Feladat: „Határozza meg és írja ki a minta szerint a sofőrök számát a forrásállományban található becenevek alapján! Feltételezheti, hogy nincs két azonos becenév.”

Feladat értelmezése: Számoljuk meg, hogy hány taxisofőr halmazt tartalmaz összesen a halmazábránk.

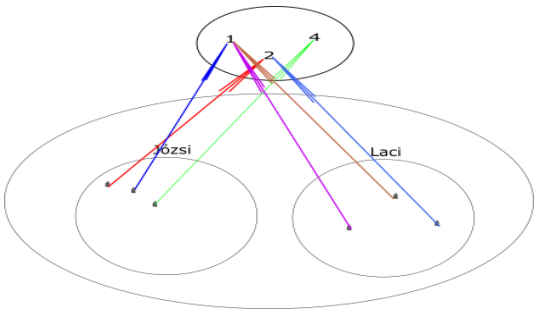


Rendezzük az ábránkat, hogy csak a szükséges adatok látszódjanak (Taxisofőrök). A keresett információ, a taxisofőrök halmaznak az elemszáma.

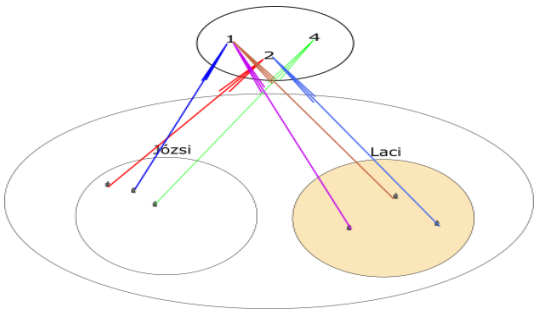
22. ábra: Taxisofőrök megszámlálása (Forrás: Saját készítés)

Feladat: „Határozza meg a legtöbb adást indító sofőr nevét! A sofőr neve és az általa indított hívások száma a minta szerint jelenen meg a képernyőn!”

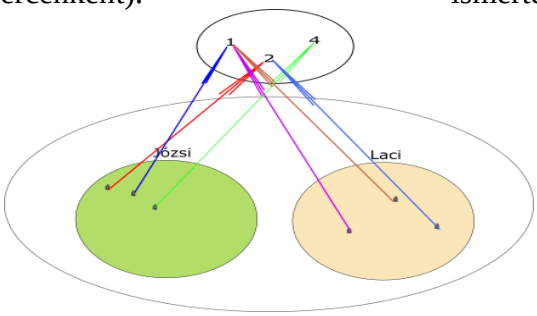
Feladat értelmezése: Összesítsük a taxisofőrökhöz tartozó hívások számát. A hívásokból könnyedén kiválaszthatjuk a maximumot a korábban már ismertetett minimumkereséshez hasonlóan. Ha végigértünk a halmazokon, ellenőrizzük, hogy az adott számadat kihez tartozik.



1. Rendezzük az ábránkat, hogy csak a szükséges adatok látszódjanak (Taxisofőrök, hívások száma percenként).



2. Válasszunk ki egy tetszőleges halmazt, és számoljuk az összes hívását, a korábban már ismertetett módon.



3. Válasszunk ki egy újabb halmazt, és annak is összegezzük a hívásai számát. A két értéket hasonlítsuk össze. A korábban már ismertetett minimum keresés algoritmushoz hasonlóan, azt az értéket fogjuk hasonlítani a taxisofőrök halmazának további elemeihez, melynek értéke nagyobb.

23. ábra: Legtöbb hívást intéző taxisofőr keresése (Forrás: Saját készítés)

4.4. Összegzés

A feladatok során halmazok segítségével ábrázoltuk a logikai műveleteket. A programozási feladatok azon részét, melyek nem a nyelvi absztrakciókra vonatkoznak, pusztán matematikai műveletekre egyszerűsítettük. Ez arra enged következtetni, hogy a programozási műveletek matematikaiakkal leképezhetőek. Ebből az következik, hogy a gondolkodási műveletek, melyeket a matematika tanulásakor elsajátítunk, hasznunkra válnak akkor, amikor programok eljárásait implementáljuk.

Személyes véleményem, hogy a matematika oktatásának megújítása közben elért eredményeket a programozás tanítása közben is fel lehet használni. A leírtakból kiindulva, a gondolkodási műveletek fejlesztésének esetében feltételezhetjük, hogy az említett két tantárgy oktatása kölcsönösen kiegészítheti egymást. Míg a matematika az absztrakciók empirikus tudománya, a programozás egy új aspektusba helyezi a már elsajátított tudást. Az alkalmazásfejlesztés során a fejlesztő kreativitása határozza meg, hogy az általa ismert logikai kapcsolatokat hogyan definiálja a feladatai megoldásában. Ennek köszönhetően az említett művelet lehet, hogy az alkotói szabadságot előtérbe helyezi, de a hátrányáról sem feledkezhetünk meg. A fogalomalkotás a programozásban nagy részben függ attól, hogy a fejlesztő milyen keretek között értelmezi a fogalmat. Ezért feltételezhetjük, hogy a fejlesztők azonos szintű matematikai ismereteinek kialakítása növeli az együttműködés hatékonyságát, mivel a magasabb absztrakciós szinteken azonos struktúrák kialakítására törekednek.

A problémamegoldás tanulásának kezdeti szakaszában rengeteg akadállyal találja szemben magát a diák. Azt gondolom, hogy ezen a szinten lehetne leginkább hasznosítani matematika keretében már elsajátított *szkémákat*. A XX. században élt matematikusok, akik célul tűzték ki azt, hogy megreformálják a matematika oktatását, egyetértettek abban, hogy a gondolati struktúrák fejlesztése többet ér bármilyen “bemagolt” értelmezhetetlen szabálynál. A fejezetben bemutatott összefüggések alapján feltételezhető, hogy a programozást fel lehetne használni a matematika tudásunk elmélyítésére, mert a gondolati struktúrák fejlesztéséhez és megszilárdításához egyaránt hozzájárul a problémamegoldás folyamata.

5. Esettanulmány az analógiák szerepéről a nyelvi elemek elsajátításában

A korábbi fejezetekben láthattuk, hogy a problémamegoldáshoz fontos az absztrachálás készségének fejlesztése, amit a programozás során is alkalmazni tudunk. A matematikai ábrák bár megkönnyítik az absztrakciók kialakítását, a sikeresebb együttműködés érdekében tantermi körülmények között más eszközöket is használhatunk, hogy a tanulók érdeklődését felkeltsük.
A korábban említettek alapján az absztrakciók (pl.: keresés algoritmus) nem csak matematikai fogalmakkal modellezhetőek le könnyedén, de hétköznapi cselekmények segítségével is. A programozás oktatása során elsősorban a probléma mögötti matematika, illetve logika megértésére fókuszáltam, hogy a tanulók megértési folyamatait támogassam.
Tanításaim alkalmával különböző életkorú és előképzettségű diákokkal találkoztam, melyek során a magyarázatokat folyamatosan finomítottam annak érdekében, hogy megfeleljenek az életkor sajátosságainak. Bár Winkler Márta könyve (Winkler 2015) alsó tagozatos tanulókkal való foglalkozásainak tapasztalatait összegzi, a hasznos gondolatait megpróbáltam beépíteni az óráimba.
Mivel az esetek többségében elvonatkoztatunk a konkrét problémától, ezért a matematikai feladatmegoldás közben fontos elem, hogy a diákok magukénak érezzék az adott feladatot, és ösztönözve érezzék magukat annak megoldására. Ez utóbbiban lehet segítségünkre a probléma megoldása során a feszültség kialakítása. A hétköznapi példák használata könnyebben megalapozza az önálló gondolatok kialakulását a tanulókban, és ez ösztönözheti őket a feladat megoldására. (Winkler 2015: 82-84)
A fent leírtak miatt a programozás oktatása során már a nyelvi elemek megismerése közben lehetőségünk van megteremteni egy olyan környezetet, ahol az alkotó gondolkodásra helyezzük a hangsúlyt. A tapasztalataim szerint minél több hétköznapi példán keresztül ismerik fel a diákok az algoritmusokat, annál biztosabban fogják azokat alkalmazni a problémamegoldás során. Kezdetben sok esetben akadályokba ütközik annak a felismerése, hogy egy feladat megoldása tartományfüggetlen, és csak az implementáció során térünk rá a programozáshoz szükséges nyelvi elemek ismertetésére. Annak érdekében, hogy ez a készség kialakuljon, a segítséget minden esetben egy másik probléma megoldásának ismertetésével biztosítottam a diákok számára. Ezáltal a tanulók csak akkor tudták felhasználni a segítségemet, ha általánosították annak a megoldási folyamatát, majd leképezték azt a konkrét feladatra.

Fontos nevelési célom is volt azzal, hogy mindig új analógiákat vezettem be a segítség nyújtásakor. Ezzel is jeleztem a diákoknak a feladatok sokszínűségét és a felismerés fontosságát, utalva arra, hogy minden problémát nem tudunk megtárgyalni a foglalkozásainkon. A diák számára nem az a fontos, hogy megtanuljon megoldani egy feladatot, sokkal inkább el kell sajátítania egy feladat megoldásához szükséges attitűdöt, amelynek kialakításában fontos szerepe van a tanárnak (Boráros-Bakucz 2018: 103-104).

A programozás tanulási folyamatának a kezdeti szakaszban egy másik fontos eleme, hogy a diákok az algoritmizálási készségeikkel különálló részfeladatokat alakítsanak ki a feladatokból. Amikor a tanulók a nyelvi elemeket még nem ismerik, és a programot a problémamegoldási folyamat részeit összevonva próbálják meg implementálni, gyakran kudarcélményekkel szembesülnek. Ennek oka, hogy egy kevésbé ismert tartomány szimbólumaival szeretnének dolgozni, amely megnehezíti a megoldás átgondolását. Emiatt a diák egyből legalább két problémával is találkozik, mert amellett, hogy nem tudja megoldani a feladatot, az implementációt sem képes elkezdeni. Ezért fontosnak tartom, hogy a tanulók megismerkedjenek a problémamegoldás menetével.

Az esettanulmányaim kiválasztásakor arra törekedtem, hogy azokat a tanítási gyakorlatom során szerzett tapasztalataimat osszam meg, melyek nagy hatással voltak a tanári jellemfejlődésemre. A technikám fejlődése és az oktatói tapasztalataim befolyásolták a tanítási folyamat eredményességét. Mivel a reflexiók függetlenek attól, hogy mikor találkoztam a problémákkal, ezért nem időrendi sorrendet követek a tanulmányok bemutatása során.

5.1. Eldöntés algoritmus

Az eldöntés algoritmus segítségével tudjuk meghatározni, hogy egy előre definiált elemeket tartalmazó struktúra rendelkezik-e olyan elemmel, amelyre egy állítás igaz. Az eljárás segítségével vizsgáljuk, hogy az elemek között szerepel-e egy bizonyos elem. (Heineman – Pollice – Selkow 2009: 106-111)

Feltételezésem szerint az algoritmus elsajátítását a definíció ismertetése nem könnyítette volna, ezért egy problémán keresztül vezettem fel. A feladat a következő volt:

„Van egy titkos csoport, ami új tagokat toboroz. Egy tagot akkor vesznek fel, ha az új tag eltalálja a jelszót. Az új tagnak a következő jelszavak valamelyikét kell ismernie: kirándulás, ecet, alma, fehér, zöld. Írjunk egy programot, ami bekér a felhasználótól egy szót! Ha a felhasználó eltalálja valamelyik jelszót, akkor írjuk ki, hogy »Gratulálunk, felvételt nyertél a

csoportunkba!«, egyébként azt jelenítsük meg a képernyőn, hogy »Sajnáljuk, a szó nem a titkos jelszavaink egyike.«"

A diákok felismerték a feladatot elemezve, hogy egy lista adatszerkezetet kell bevezetniük az implementációhoz, ami tartalmazza a felsorolt lehetséges jelszavakat. Sajnálatos módon a kezdeti siker miatt, a feladat átgondolása nélkül álltak neki a tanulók az algoritmus megvalósításnak, ahol el is akadtak. A probléma megoldásának kulcsa ugyanis, hogy felismerjük az ismétlődő tevékenységet, ami azonban a példán keresztül nem triviális. Több diák a félig megírt kódrészletet mutatva kért tőlem segítséget, annak ellenére, hogy a közös tervezés fázisában voltunk még. Azt gondolom, hogy ők tévesen a kódolást tekintették az egyetlen és megbonthatatlan alkotói tevékenységnek. Ez a magatartás eddigi tapasztalataim szerint általános, ezért a tanítási gyakorlatom megkezdésétől próbálom ösztönözni a diákokat a több fázisból álló problémamegoldási folyamatok kialakítására.

Ebben az esetben az ismételt tevékenységre próbáltam felhívni hallgatóság figyelmét, ám a feladat komplexitása miatt ezt nem tehettem meg a példánkat felhasználva. Ha mégis így jártam volna el, akkor megoldottam volna a feladatot a diákok helyett, ezzel megfosztva őket az alkotás közben átélhető örömtől. Végül a következő párbeszéd segített minket a probléma megoldásában:

Tanár: Vonatkoztassunk el a feladattól. Tegyük fel, hogy elvesztettétek otthon a lakáskulcsotokat. Hogyan találhatjátok meg?

Diákok: Visszaemlékezek, hogy hol vesztettem el.

Tanár: Ha nem jut eszetekbe, hogy hol vesztettétek el, akkor mit csináltok?

Diákok: Elindulok megkeresni.

Tanár: Hogy kezditek a keresést?

Diákok: Átnézem a szobámat.

Tanár: Ha ott nincs, akkor mit csináltok?

Diákok: Keresem egy másik szobában.

Ennél a pontnál úgy éreztem, hogy az analógia bevezetése sikeres volt, már csak néhány kérdéssel rá kellett mutatnom a feladat és a hétköznapi cselekmények közötti párhuzamra.

Tanár: Mit fogtok csinálni, ha abban a szobában sincs meg?

Diákok: Folytatom a keresést.

Tanár: Meddig folytatjátok a keresést?

Diákok: Amíg át nem néztem minden szobát.

Tanár: Mi az ismételt tevékenység a példánkban?

Diákok: Átmegyek a másik szobába és keresem a kulcsot.

Bár a feladat megoldásától most távolabb kerültünk, felismertük az ismételt tevékenységet, és azt, hogy szobáink száma befolyásolja a keresésünk hosszát. Ez az eredeti feladat megoldásában a szavak számával állítható párhuzamba.

Tanár: Ha megtaláltátok a kulcsotokat, akkor folytatjátok-e a keresést? Honnan tudjátok, hogy megtaláltátok a kulcsotokat?

Diákok: Ha megtalálom kulcsot, akkor nem folytatom a keresést. Onnan tudom, hogy meglett a kulcsom, és nálam van.

Tanár: Azt felfoghatjuk egy vizsgálatként, hogy megnézed nálad van-e a kulcs?

Diákok: Igen.

Tanár: Ha ezt is mindig el kell végeznünk, akkor hogy módosul az ismételt tevékenységeink listája?

Diákok: Átmegyek másik szobába, keresem a kulcsot, és ha megvan a kulcs, akkor abbahagyom a keresést.

Ezen a ponton kijelenthetjük, hogy a diákok kimondták az eldöntés algoritmusát. Már csak egy dolog maradt hátra, hogy összeszedjük a lépéseket.

Tanár: Hogy hangzik a kulcskeresésünk algoritmusa egyben?

Diákok: Végigmegyünk a szobákon, és minden szobában megvizsgáljuk, hogy megtaláltuk-e a kulcsot. Ha nem találtuk meg, akkor továbbmegyünk a következő szobába.

Miután megoldottuk ezt a hétköznapi példát, még annyi feladatunk maradt, hogy megállapítsuk az egyes lépések közötti párhuzamot.

Tanár: Az eredeti példánkban hogyan tudnánk ezt az algoritmust hasznosítani?

Diákok: A jelszavak listája az olyasmi, mint a szobák listája. A vizsgálat meg lehet akár a bekért szöveg és az aktuális jelszó összehasonlítása is.

A párbeszéd végére eljutottunk egy olyan megoldáshoz, amelyet fel tudnak használni a diákok az eredeti feladat megoldására is. A tanulók az implementáció előtt értelmezték a szükséges algoritmust, így a megvalósításhoz már csak a programnyelv elemeit kellett ismerniük.

A magyarázat hasznosságát jól szemlélteti, hogy a diákok többségének az algoritmizálás nem okozott gondot, és csak pár nyelvi elemet kellett átismételni a programkódok javítása során.

5.2. Objektum Orientált Programozás alapjai

Az objektum orientált programozás tanítási céljaként azt határoztam meg, hogy a tanulási folyamat végére a tanuló önállóan felismerje *tervezési mintákat* (Gamma-Helm-Johnson-

Vlissides 1994). Ezek a struktúrák elengedhetetlenek a szoftverfejlesztéshez. Egy alkalmazás fejlesztése közben a programozónak fel kell ismernie a korábban már implementált mintákat, valamint a feladatait úgy kell elvégeznie, hogy az általa megírt programkód illeszkedjen azokhoz.

Az objektum orientált programozáshoz egy olyan projektet állítottam össze, ahol a tanuló gyakorolhatja az osztályhierarchiákat. A cél egy olyan alkalmazás fejlesztése volt, ahol a felhasználó nyilvántarthatja a könyveit. Az implementációhoz biztosítottam a diák számára, hogy milyen struktúrákat szeretnénk kezelni, ezeket hogyan tudjuk meghatározni a programozási nyelvben, valamint részekre bontottam a megvalósítást. Az utóbbit azért éreztem szükségesnek, mert úgy ítéltem meg egy előzetes felmérésem segítségével, hogy a diák korábban még nem találkozott ilyen komplex feladattal.

Minden rész során arra törekedtem, hogy mindig csak egy új strukturális elem kerüljön bevezetésre. Az elérhető források mellé én is létrehoztam a saját segédanyagomat. A célom az volt, hogy hasonló implementációk segítségével példát biztosítsak tanuló számára. Az elkészült alkalmazásomat ugyanúgy lépésekre bontottam, és a projektünk implementációjának részegységeihez rendeltem. Magyarázatom során kitértem az egyes fázisok közötti különbségekre, amelyekről külön jegyzetet is készítettem a diák számára. A célom egy olyan analógia biztosítása volt, aminek a feldolgozása segíti a tanulót a projekt feladatunk megoldásában.

A példa alkalmazásom egy madarakat rendszerező szoftver volt. Az egyes osztályok létrehozásakor a tulajdonságokat szubjektív mérce alapján határoztam meg. A példa egyszerűsége miatt megsértettem a konzisztenciát az egyes osztályok esetében, mert nem következtethettünk egy adott lépés során arra, hogy miért kell írnunk bizonyos kódrészleteket a programba. Magyarázatként definiáltam egy Madár osztályt, amelynek két leszármazottja van: Pingvin és Holló. A Pingvin osztály példányai tudtak úszni, míg a Holló egyedei repülni. Ezek olyan evidenciák, amiket nem indokolt a probléma leírása.

Egy másik hiba a segédanyagomban, hogy a Singleton és a Composite (Gamma-Helm-Johnson-Vlissides 1994: 144-152, 183-196) *tervezési minták* használatára egy-egy példát vezettem be, ám a diáknak ezeket struktúrákat több helyen is implementálnia kellett a projekt megoldása során. A minták használatát tanárként elvártam a tanulótól, viszont azok nélkül is megoldható lett volna a projektfeladatunk.

Ennek köszönhetően a diák arra a következtetésre jutott, hogy a *tervezési minták* és az osztályok egyéni igények mentén kerülnek definiálásra. Bár a megállapítás részben helyes, ez gátolta

példakódom megértését, melynek eredményeként egy olyan absztrakció megállapítását vártam, amelyet a projektfeladat során alkalmazni tud a tanuló.
A projekt közben a példakód és a források segítségével a tanuló elsajátította a tartományspecifikus elemeket. Ez részben annak köszönhető, hogy a diák a kódolás közben találkozott a szükséges nyelvi elemekkel. Azonban a *tervezési minták* és az összetett struktúrák alkalmazása a projekt megvalósítása után is akadályokba ütközött. Ezen segíthetne a jövőben, ha kevesebb szubjektív meghatározást használnék a példáim és a tanári magyarázatom során.

5.3. Számlálás algoritmus

A számlálás algoritmusa során egy több elemből álló struktúrában számoljuk meg azokat az elemeket, amelyekre egy feltétel igaz (Szlávi – Törley – Zsakó 2019: 2-3). Ha például az 1 és 10 közötti páros számokat számlálnánk meg, akkor az említett eljárásra van szükségünk.
Az implementáció előtt a diákokkal közösen egy szimulációt végeztünk. A feladat az volt, hogy egy csomagban található szaloncukrokat számoljunk össze. A számosságát előre nem ismertük az elemeknek, ezért a probléma megoldásának érdekében egy algoritmust definiáltunk. A feladatban kikötöttem, hogy a számlálást három diák közös munkájával kell elvégezni. Ha csak egy diák végezte volna el a feladatot, akkor az algoritmus egyes lépései nem különültek volna el egymástól, így nehezebb lett volna azt az absztrakciót átgondolni, amit a későbbiekben implementálni szerettünk volna.
A három diák különböző feladatokat végzett. Az egyik tanuló megkérdezte a másikat, aki a csomag szaloncukornál állt, hogy „Van még szaloncukor?”. Az utóbbi diák igennel vagy nemmel válaszolt, majd kivett egy szaloncukrot a csomagból. Ha a kérdésre adott válasz „Igen” volt, akkor a harmadik tanuló húzott egy függőleges vonalat a táblára. A feladat akkor ért véget, ha elfogyott az összes szaloncukor a csomagból. Amikor a szaloncukroknál álló diák nemmel válaszolt, akkor a táblánál álló diák összeszámolta a táblán lévő függőleges vonalakat.
A gyakorlat után a csoporttal közösen kielemeztük a történteket. Először felírtuk az eljátszott algoritmust, majd utána pszeudokódban is implementáltuk az általános számlálás algoritmust. Bár a klasszikus eljárás tartalmaz további feltételt is, amely az elemek tulajdonságára vonatkozik, én az első feladat esetében nem szándékoztam szűrni az elemeket, de a pszeudokód tárgyalása közben kitértem rá. További feltétel bevezetése is lehetséges lett volna, ám az komplexebb instrukciókat eredményezhet. Több diák bevonásával tovább bonyolíthatóak az

algoritmusok, de érdemes azt szem előtt tartani, hogy az eljárás instrukciói még könnyen érthetőek kell legyenek a tanulócsoport nem aktívan közreműködő tagjai számára is.
Miután az absztrakció felírásáig eljutottunk, megkértem a diákokat, hogy implementálják a korábban eljátszott programot azzal a módosítással, hogy felhasználótól bekért válasz alapján növeljünk egy változót, vagy írjuk ki az értékét. A program megvalósítása nem okozott problémát a diákok számára, ebből arra következtettem, hogy az elmélet megértését segítette a közös játék.
Mivel a játékos felvezetés egy program struktúráját követi, ahol az utasítások egymás után következnek, ezért azt gondolom, hogy a jövőben érdemes komplexebb algoritmusoknál is alkalmazni ezt az eszközt, akár több tanuló bevonásával is. Ha a programozás tartományán belül keressük a játék további alkalmazási lehetőségeit, akkor a *clean architecture* (Martin 2017) lemodellezésére is alkalmas lehet tantermi körülmények között.

5.4. Tippelős játék

A programozási tanulmányok kezdetén a diákok a ciklus alkalmazásával nem mindig vannak tisztában. Annak érdekében, hogy ezeknek a vezérlési szerkezeteknek a működését megértsük, több algoritmust is értelmezünk közösen. Az eljárások értelmezésekor érdemes azt is megemlíteni, hogy ha segédváltozókat használunk, akkor azok értékeit mikor változtatjuk vagy vizsgáljuk.
Az említett probléma megértésére a következő feladatot adtam fel egy tanulócsoportnak: *„Generáljunk egy véletlen számot 1 és 10 között. A felhasználó háromszor próbálkozhat a generált szám kitalálásával. Ha nem sikerült eltalálnia a számot, akkor írjuk ki a következőt: »Sajnos nem nyert«. Ellenkező esetben gratuláljunk a felhasználónak akkor, amikor eltalálta a számot.”*
A véletlen szám generálása egy korábbi órának az anyaga volt, így a folyamat értelmezésétől eltekintettem a magyarázatom során. Azok a tanulók, akik rögtön elkezdték a program implementációját, hamar szembesültek a tervezés hiányának következményeivel, mert csak a véletlen szám generálásáig jutottak. A tervezés megkönnyítésének érdekében, ebben az esetben is egy hétköznapi analógiát vezettünk be közösen a tanulókkal.
Itt először elmondtam egy magyarázatot a tanulócsoportnak, majd megkértem a diákokat, hogy a feladat megoldásának implementálása után magyarázzák el az párhuzamokat a két feladat egy-egy eleme között. A magyarázatom így hangzott:

„Van egy kulcskarikád a zsebedben, de elveszett a kulcsod. Szobáról szobára jársz és keresed a kulcsodat. Ha megtaláltad a kulcsodat, akkor ráteszed a kulcskarikádra, és nem folytatod tovább a keresést. Miután végigértél az összes szobán, megvizsgálod a kulcskarikádat. Ekkor megállapítod, hogy az még mindig üres, ebben az esetben a keresésed sikertelen volt."

A magyarázatom után a diákok önállóan implementálták a programot egy programozási nyelvben. Miután végeztek, az egyik önként jelentkező így magyarázta a párhuzamokat a kulcskeresés és tippelős játék megoldásai között:

„A kulcskarika egy logikai változó ebben az esetben. Van egy ciklusunk, ami a feladatnál a tippelések száma, a példában pedig a szobák száma. Ha megtaláltuk a kulcsot, akkor feltesszük a kulcskarikára. Ez a programunkban a logikai változó értékének változása. Mivel megtaláltuk a kulcsot, nem folytatjuk a keresést, tehát itt ki kell írnunk a »Gratulálok« szöveget, majd meg kell szakítani a ciklusunk futását egy paranccsal. A keresés után megvizsgáljuk újra kulcskarikát, ami a feladatunkban a logikai változónk értékének vizsgálata. Ha a kulcskarikánk üres, vagy a logikai változónk nem változott, akkor kiírjuk, hogy »Sajnos nem nyert«"

Ha a feladat megoldásának programkódjait értelmeztük volna, akkor többféle elkészült kód elemzését is el kellett volna végeznünk. Erre példa, hogy volt olyan tanuló, aki a logikai változó kezdeti értékének „igaz", míg mások „hamis" értéket adták. Ha a kódot nézzük, akkor mindkét implementáció működhet helyesen, ha a logikai vizsgálatok jók. Nem mondhatjuk azt hibának, ha a megoldás helyes, csak más a logika. Az egy magasabb fokú absztrakció, amit a diák megállapított: *„a logikai változónk nem változott"*. Azt az értékelésnél figyelembe kell vennünk, hogy a feltételt úgy írta-e fel tanuló, ahogyan azt a változót használja az algoritmusban. A diákok magyarázatai itt sokat segíthetnek a tanárnak, mert a programkódból nehéz megítélni, hogy mi volt az alapja az implementált logikának.

Az eset után elhatároztam, hogy a jövőben célként fogom definiálni, hogy a diákok saját analógiák alapján tervezzék meg egy adott probléma megoldását. Emiatt tanárként kezdetben törekednem kell arra, hogy elősegítsem annak a rutinnak a kialakulását, hogy a diákok megpróbálják egy absztrakt platformra áthelyezni a problémát a megoldás előtt, ahogyan azt korábban én biztosítottam számukra. Ennek érdekében mindig megvárom, amíg saját szavakkal megbeszéljük a probléma megoldását a diákokkal, és ha logikai hiba van, akkor sem a programkódban javítunk közvetlenül. A kérdéseket ezáltal első körben nem az implementációhoz kapcsolódóan, hanem a tervre vonatkozóan teszem fel, ha elakad egy diák a feladat megoldásában. Gyakran ez már segíthet annyit, hogy a tanuló önállóan be tudja fejezni a feladatát. A ritkább esetek közé tartozik, ha a megoldás folyamatát programozási nyelv ismeretének hiánya is gátolja.

5.5. Fájlolvasás

A fájlolvasás művelete csak a programozás tartományán belül értelmezhető, és rendelkezik olyan nyelvspecifikus elemekkel, amelyek az implementáció nyelvéhez köthetőek. Az absztrakció megértése tehát ebben az esetben elengedhetetlen, melynek segítségével a különböző programnyelveknél a tanulónak nem lesz szüksége az eljárás ismételt elsajátítására. A nyelvspecifikus eszközök ismeretére a fájlolvasáshoz köthető problémák megoldásához továbbra is szüksége lesz a programozónak, de az egyes utasítások csak elnevezésükben különböznek. További eltérés lehet a programozási nyelvek között, hogy néha tartalmaznak tömörebb utasításokat, melyek segítségével több utasítást vonhatunk össze. Kezdő programozók esetében úgy gondolom, hogy ezek a „könnyítések" megnehezítik az eljárások megértését csakúgy, mint a matematika esetében a leegyszerűsített képletek.

Mivel egy programozó gyakran találkozhat számára ismeretlen kóddal, amit önállóan kell feldolgoznia, ezért úgy határoztam a tanítási gyakorlatom során, hogy először megosztom a feladatot és annak megoldását, majd közösen sorról-sorra elemezzük azt, hétköznapi példán keresztül szemléltetve az egyes sorok működését. A feladat a következő volt: *„Olvassuk be a »szamok.txt« állományt, melynek sorai egy-egy számot tartalmaznak. Adjuk össze a fájlban található számokat, és írjuk ki az összegét!"*

A feladat szövegéből könnyen kivehető, hogy egy összegzés algoritmussal (Szlávi – Törley – Zsakó 2019: 1-2) kell megoldanunk a problémát. Azonban nem elég az összegzés eljárását alkalmazni a feladat megoldásához. Miután az implementációt megosztottam a diákokkal, páran felismerték az összegzésnek azt a jellegzetes sorát, ahol az egyes elemeket egy segédváltozóhoz adjuk hozzá, majd az eljárás végén kiírjuk a végösszeget. A programozási nyelv fájlolvasást megkönnyítő eszközeit a következő módon vezettem fel:

„Fájlolvasásnál először az állományt be kell töltenünk a memóriába. Ennél a lépésnél még nem tudjuk, hogy mit tartalmaz a fájl. Ez olyan, mintha egy sötét szobában lennénk, és egy zacskó cukrot egy tálba öntenénk. A fájlt soronként fogjuk beolvasni, mielőtt beolvasnánk meg kell győződnünk arról, hogy a fájlnak van-e még sora. Hasonlóan, ha meg akarjuk vizsgálni a cukorkákat, azokat csak egyesével tudjuk elemezni egy zseblámpa segítségével. Ha kiürült már a tál, akkor nem tudunk kivenni több cukrot belőle.

Amikor fájlból olvasunk, minden sor szöveg formátumú, ezekkel nem tudunk számolni, csak ha számmá alakítjuk őket. A cukorkás tál esetében, ha meg akarjuk vizsgálni, hogy milyen színű cukrok vannak, akkor a sötétben kivett cukorkákat a megvilágításuk után egy színnek feleltetjük meg. Az új színeket minden esetben felírjuk egy lapra. Ezt a műveletet a fájlolvasás esetében a

számmá alakított szövegnek a segédváltozóhoz való hozzáadásának tekintjük.
Az ismételt tevékenység mindkettő esetben akkor ér véget, ha már nem tudunk új adatot beolvasni, tehát vagy nincsen már új sora a fájlunknak, vagy kiürült a cukorkás tál. Ha végeztünk, akkor az olvasás után a fájlt be kell zárni, hogy más folyamatok is hozzáférjenek. A cukorkás példa esetében ez azt jelenti, hogy amennyiben kiürült a cukorkás tálunk, akkor már nincs rá szükségünk, tehát azt félretehetjük.
A két eljárásunk befejezése is azonos. Egyik esetben megjelenítjük a segédváltozónk értékét, a hétköznapi példánkban pedig megnézzük, hogy milyen színeket írtunk fel a lapra."
Az analógia ismertetése után a diákok a saját szavaikkal készíthettek jegyzetet a korábban megosztott programkód egyes soraihoz. Annak ellenére, hogy a magyarázatom egy-két esetben eltért az eredeti feladattól, a későbbiekben a diákok többsége hibátlanul alkalmazta az eljárást. Ha figyelünk a diákok visszajelzéseire és az általuk megfogalmazott példákra, akkor azt gondolom, hogy ezek az analógiák tovább tökéletesíthetőek. Ennek érdekében mindig megkérem a diákokat, hogy írjanak ők is a problémához hasonló példákat. Ha a speciális megoldásukat ilyen módon egy másik probléma megoldására is fel tudják használni, akkor azzal is fejlődik az absztrakciós készségük.
Néha a diákok a megoldásuk magyarázata helyett az egyes programsorok működését írták le. Ezzel fejlődhet az adott programnyelv, valamint a szaknyelv ismerete, de a problémamegoldási készségeikről nem árul el információt. Ezért kérni szoktam, hogy a tanulók magyarázatot is írjanak a jegyzeteikbe, és ne csak tartományspecifikus információkat. Így elkerülhetjük, hogy a tanulás folyamatában a megértést akadályozza a szaknyelven dokumentált kód. Természetesen egy programozó számára elengedhetetlen a szaknyelv ismerete, de a problémamegoldáshoz már nem feltétlenül szükséges.

5.6. Összegzés

A programozás oktatás kezdeti szakaszában két területet kell párhuzamosan fejlesztenünk. Az egyik terület a problémamegoldáshoz köthető, mely során a matematikából már ismert absztrakciókat felhasználjuk az algoritmusunk megtervezéséhez. Ebben a részben nem kell használni a programozás tartományában definiált szimbólumrendszert. Az algoritmusok megértését könnyíti, hogy korábbi tapasztalatokra építve tudjuk bővíteni a tanulók problémamegoldáshoz szükséges eszköztárát.
A programozás oktatás kezdeti szakaszának másik fontos területe a programozás

tartományában értelmezhető. Itt a matematikai struktúrákat és a tervezés során létrejött eljárásokat implementáljuk egy formális nyelvben. Az eredmény lehet pszeudokód, folyamatábra, vagy akár egy programozási nyelv is.

A tanulók által átélhető kudarcélmény lehetőségének kiküszöbölése érdekében az oktatási gyakorlatom során különválasztottam ezeket a részeket. Az elméleti részeket fokozatosan vezettem be, amelyeket a diákok használni tudtak a tervezés fázisában. A tanulócsoportok azon tagjai, akik már rendelkeztek előzetes ismeretekkel, bár a feladataik megoldásait hamarabb kitalálták, de a magyarázat során ők is rákényszerültek a probléma megoldásához szükséges műveletek átgondolására. A tapasztaltabb diákok tervezési folyamatainak közös elemzésével a kevesebb ismerettel rendelkezők is további példákat kaptak, amelyeknek segítségével az elméleti tudásukat bővíthették.

Értékeléseim során igyekeztem szűrni a programozási nyelv ismeretéből fakadó hiányosságokat azáltal, hogy esszé jellegű leírást is kértem az elkészült programkód mellé. A programozói pályán a gondolatok kifejezése rendkívül fontos, amikor az algoritmusok, illetve az alkalmazások magyarázatát kell biztosítani a tapasztaltabb fejlesztőknek, vagy a megrendelőknek. Észrevételeim szerint az a diák, aki a tervezés során a gondolatait nem tudta leírni, annak az implementációjában is voltak logikai hibák. Ennek oka a már korábban tárgyalt *szkémákban* keresendő. Ha egy szimbólumrendszer nem kapcsolódik a korábbi tapasztalatainkhoz, akkor valószínűsíthető, hogy a tudásunk alkalmazása során csak a megjegyzett, de meg nem értett műveletekkel próbálunk eredményeket elérni.

Egy programozási környezetben rendelkezünk olyan eszközökkel, amelyek megkönnyíthetik a logikai kapcsolatok megértését. Ha a tanuló csak a fejlesztőkörnyezetben gondolja át a feladat megoldását, akkor könnyen arra a következtetésre juthat, hogy a tesztelés a fejlesztés egyik utolsó fázisa. Amennyiben a problémamegoldás folyamatát nem tekintjük a programozás részének, valóban igaz az előbbi állításunk. Ha úgy fejlesztünk alkalmazást, hogy az azon ritka esetek közé tartozik, amikor logikai- és matematikai tudásra sincs szükségünk, akkor előző állításunk valóban igaznak bizonyulhat. Ilyenkor a formális nyelv elemeit kell csak ismernünk, ám ha ezek egy rendszer részeit alkotják, akkor szükségszerű kibővítenünk a tudásunkat annak a logikai struktúráival. Implementációnk tervezésének részét képezi az előbb említettek feltérképezése, így tervünk tesztelése az elkészült programkódnak is részben megfeleltethető.

A tanításaim során arra fókuszáltam, hogy a szoftverfejlesztés folyamatát és szituációit megismerjék a diákok. A magyarázataimmal és példáimmal előkészítettem a csoportjaimnak olyan helyzeteket, amelyekben egyszerűbb célokat kellett elérniük. A jegyzet készítése a programkód alapján kezdetben egyszerűbb, ezzel az implementáció sikertelensége nem merül

fel kockázatként a tanulónál. Bár az önálló munkához elengedhetetlen, ennek a műveletnek az értelmét csak később ismerik fel diákok, mert a tanulási folyamat elején még megjegyezhető mennyiségű eszköz áll a rendelkezésükre. Ha a tanulók már rutinszerűen tudják a dokumentálást végezni, akkor az megalapozhatja a későbbi tartományspecifikus eszközök megismerését is.

A programozás tanításának gyakorlatát úgy terveztem meg, hogy a foglalkozásaim során a problémamegoldás folyamataira koncentráltam, ezzel egy olyan környezetet teremtve, ahol a diákok folyamatosan nehezedő feladatokon keresztül szerzik meg azokat a tapasztalatokat, amelyeket később az önálló feladataik megoldásaiban hasznosítani tudnak. Azt gondolom, hogy ezzel egy olyan általános tudást is megszerezhetnek a tanulók, ami a programozás tartományán kívül is hasznukra válhat.

6. Összefoglalás

A dolgozatomban körüljártam, hogy a matematikai ismeretek hogyan segíthetik a problémamegoldást a programozás közben. Az analógiák használata az oktatásban megkönnyíti azoknak az általános gondolati struktúráknak a kialakulását, amelyeket a tanulók a későbbiekben alkalmazni tudnak akár a szoftverfejlesztés tartományától függetlenül is. A tervezés és megvalósítás folyamatának megismertetése azért kiemelt fontosságú, hogy a diákok ne vonják össze a lépéseket a problémamegoldás során, mert az a későbbiekben – akár más tantárgyak esetében is – megnehezíti a sikeres feladatmegoldást.

Ezt követően megvizsgáltam a Nemzeti Alaptanterv (NAT 2020) követelményrendszerét, ahol összefüggéseket kerestem arra, hogy a kompetenciákra vonatkozó célkitűzések hogyan járulnak hozzá a problémamegoldáshoz szükséges készségek fejlesztéséhez. A célokat csoportosítottam az alapján, hogy az említett területhez általánosan járulnak-e hozzá absztrakt eszközökkel, vagy tartományspecifikus tudással támogatják azt. Megfigyeltem a matematika és a digitális kultúra tantárgy elemzésekor azokat a tételeket, amelyek segíthetik a diákokat a számítógépes környezetben történő problémamegoldási gyakorlat kialakításában. Bár a dolgozat terjedelmi korlátai miatt erre nem került sor, érdemes volna megvizsgálni a rendelkezésre bocsátott digitális tananyagokat a különböző taxonómia mátrixok alapján.

A következő fejezetben elemeztem a matematikai gondolkodás pszichológiai hátterét, valamint azokat a motivációs tényezőket, amelyeket érdemes figyelembe venni a tanítási gyakorlat során.

A dolgozatom következő részét képező fejlesztési részben a matematikai fogalmak felhasználásával definiáltam egy olyan absztrakt platformot, amelyet több algoritmus megoldása során alkalmaztam A platform alkalmazása során új analógiákat hoztunk létre a feladatok megoldásainak megértése érdekében. A pszichológiai háttér áttekintését a jövőben érdemes lenne kiegészíteni kognitív tudományok különböző releváns elméleteivel.

A kutatásom gyakorlati alkalmazása során az analógiákkal szemléltettem a problémamegoldás folyamatát. Az oktatás közben a diákokkal együttműködve bontakozott ki egy olyan kollektív gondolkodási folyamat, melynek során közösen érhettünk el sikereket a feladatmegoldások implementációinak elemzése nélkül. Ugyan nem tértem ki rá, érdemes lehet megvizsgálni a gyakorlatban, hogy a diákok különböző szerepkörbe való belehelyezése egy-egy analógia alkalmazása során mennyire járul hozzá a flow-élmény kialakulásához.

További kutatás tárgyát képezheti a dolgozatban kifejtett gondolkodási struktúrák és az OOP osztályhierarchiája közötti összefüggések vizsgálata. Feltételezésem szerint az objektum orientált alkalmazás tervezése és a matematikai megértés folyamata átfedéseket tartalmaz.

Ugyan jelen kutatás nem tárgyalja, azonban a jövőben érdemes lenne megvizsgálni a bemutatott módszer integrálhatóságát a probléma alapú tanulás módszertanába. A bemutatott absztrakt platformból kiindulva érdemes volna megvizsgálni azt is, hogy további matematikai fogalmakat hogyan volna lehetséges beépíteni a programozás oktatásba.

7. Conclusion

In my thesis, I have explored how mathematical knowledge can help us with problem solving during the act of programming. The use of analogies facilitates the teaching of the general structures of thought during the education and the students can apply those later on, even beyond the domain of sopftware development. The process of design and implementation is important to avoid students confusing the steps in problem solving, which can make it difficult to solve problems successfully later, even in other subjects.

I then examined the National Core Curriculum (NAT 2020) framework of requirements, and I was looking for correlations on how objectives of competences contribute to the development of problem-solving skills. I have grouped those objectives according to whether they contribute to the domain in general with abstract tools or support it with domain specific knowledge. I have observed the statements during the analysis of mathematics and digital literacy that can help develop the problem solving practice for students in a computer environment. Although this was not done due to the scope limitations of the thesis, it would be worthwhile to examine the digital learning materials provided against the different taxonomies.

In the next chapter, I have analysed the psychological background behind mathematical thinking and the motivational factors that are very important in the teaching practice. In the development part, which is the next part of my thesis, I defined an abstract platform using mathematical concepts and I used it for solving several algorithms. In the future, it might be worthwhile to extend the overview of psychological background the different relevant theories from cognitive sciences.

In the practical application of my research, I used the analogies to illustrate the process of problem solving. In the process of teaching, a collective thinking process unfolded in collaboration with the students, whereby we achieved success together without analysing the implementation of their solutions. Although I did not discuss it, it may be worthwhile to investigate in practice how the placement of students in different roles during the implementation of an analogy contributes to the flow experience.

Further research could be undertaken to investigate the relationship between the cognitive structures discussed in this thesis and the OOP class hierarchy. My hypotesis is that the process of designing an object-oriented application overlaps the process of mathematical understanding. Although not discussed in this research, it would be wortwhile to investigate in the future how the presented method can be integrated into the methodology of problem-based learning.

Starting from the presented abstract platform, it would also be worthwhile to investigate how additional mathematical concepts could be integrated into programming education.

Felhasznált irodalom

Ballér Endre (2001): Új tendenciák a tantervelméletben és a tantervfejlesztésben. *Iskolakultúra, 11. évf. 9. szám*
Elektronikus forrás: http://real.mtak.hu/61022/1/EPA00011_iskolakultura_2001_09_067-072.pdf
Megtekintés dátuma: 2022.01.29

Bárdossy Ildikó (2006): A curriculumfejlesztés elméleti és gyakorlati kérdései. In: Bárdossy Ildikó – Forray R. Katalin – Kéri Katalin (szerk.): *Tananyagok a pedagógiai szakos alapképzéshez.* PTE BTK Neveléstudományi Intézet. Pécs.
Elektronikus forrás: https://mek.oszk.hu/15600/15612/pdf/hefoptotal.pdf
Megtekintés dátuma: 2022.01.21

Bárdossy Ildikó (2011): A curriculumelmélet értelmezései, kutatási irányai. In: *Lehetséges kérdések és válaszok a curriculumfejlesztéshez – Tananyag egyetemi hallgatók és pedagógusok számára.* Pécsi Tudományegyetem. Pécs
Elektronikus forrás:
http://janus.ttk.pte.hu/tamop/tananyagok/curriculum/i_2__a_curriculumelmlet_rtelmezsei_kutatsi_irnyai.html
Megtekintés dátuma: 2022.01.27

Boráros-Bakucz András (2018): Ki mentse meg a világot? Feladata-e egy ma pályakezdő tanárnak, hogy felkészítse a jövő nemzedékének hőseit a világ megmentésére? In: Nagy Ádám (szerk.): *Nevelj Jedit! A képzelet pedagógiája.* Athenaeum Kiadó

Csernai Zoltán (2019): Az informatikai gondolkodás (computational thinking) fogalmi keretei. NETWORKSHOP, Magyarország, dec. 2019.
Elektronikus forrás: http://ocs.mtak.hu/index.php/nws/2019/paper/view/40
Megtekintés dátuma: 2021.02.20

Dienes Zoltán Pál (2015): *Építsük fel a matematikát.* Edge 2000 kiadó. Budapest.

Dienes Zoltán Pál (2018): *Agykalandok – Dienes professzor játékai.* Edge 2000 kiadó. Budapest.

Gamma, Erich – Helm, Richard – Johnson, Ralph – Vlissides, John (1994): Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional

Gaskó Krisztina (2006): A tanulás pszichológiai értelmezése. In: M. Nádasdi Mária (szerk.): *A gyakorlati pedagógia néhány alapkérdése*. ELTE PPK Neveléstudományi Intézet. Budapest.
Elektronikus forrás:
http://gepeskonyv.btk.elte.hu/adatok/Pedagogia/84N%e1dasi/Nyomtat%e1sra/pdf/Hat%e9kony%20tanul%e1s.pdf
Megtekintés dátuma: 2021.02.20

Greenfox (2022): Milyen tudást szerezhetsz meg? Greenfox academy.
Internetes elérhetőség: https://www.greenfoxacademy.com/junior-programozo-kepzeseink
letöltés dátuma: 2022.01.18.

Hazai bootcampek (2022): Néhány hazai példa a bootcampekre.
Internetes elérhetőség: https://www.greenfoxacademy.com/ https://codecool.com/hu/ https://webuni.hu/

Heiman, T. George – Pollice, Gary – Selkow, Stanley (2009): *Algorithms in a Nutshell.* O'reilly

Józsa Krisztián (2002a): Tanulási motiváció és humán műveltség. In: Csapó Benő (szerk.): *Az iskolai műveltség.* Osiris kiadó. Budapest.
Elektronikus forrás:
https://www.researchgate.net/publication/305332319_Tanulasi_motivacio_es_human_muveltseg
Megtekintés dátuma: 2021.02.20

Józsa Krisztián (2002b): Az elsajátítási motiváció pedagógiai jelentősége. *Magyar pedagógia.* 102. 1.sz.

Elektronikus forrás:
https://www.researchgate.net/publication/255612140_AZ_ELSAJATITASI_MOTIVACIO_PEDAGOGIAI_JELENTOSEGE
Megtekintés dátuma: 2021.02.20

Kátai Zoltán (2020): Algoritmustervezési stratégiák. Scientia Kiadó. Kolozsvár.
Elektronikus forrás:
http://real.mtak.hu/116698/1/Katai%20Zoltan%20Algoritmustervezesi%20strategiak%20REAL.pdf
Megtekintés dátuma: 2021.02.20

Kő Natasa – Pajor Gabriella – Szabó Mónika (2017): Motiváció. In: N. Kollár Katalin, Szabó Éva (szerk.): *Pedagógusok pszichológiai kézikönyve I. Kötet.* Osiris Kiadó. Budapest.
Elektronikus forrás:
https://dtk.tankonyvtar.hu/bitstream/handle/123456789/2913/pszichoped_1.pdf?sequence=1&isAllowed=y
Megtekintés dátuma: 2021.02.20

KKK (2020): Képzési és kimeneti követelmények Szoftverfejlesztő és -tesztelő szakmához
Elektronikus forrás:
https://api.ikk.hu/storage/uploads/files/kkk_informatika_szoftverfejleszto_es_tesztelo_tech_2020pdf-1589880410952.pdf
Megtekintés dátuma: 2022.01.20

Klein Sándor – Greer, Brian – Zétényi Tamás (2020): Halmazműveletek tanulásának fejlődéslélektani vizsgálata. In: Klein Sándor (szerk.): *Intelligencia, kreativitás, kompetencia – Módszerek és eredmények*. Edge 2000 kiadó. Budapest.

Lakatos Imre (2021): *A gyakorló matematikus filozófiája.* Typotex Kiadó

Molnár Adrienn – Fodor Szilvia – Kurucz Győző (2020): A matematikai szorongás vizsgálata a célorientációs elmélet keretében. *Alkalmazott pszichológia* 20. 1. sz.
Elektronikus forrás: http://ap.elte.hu/wp-content/uploads/2020/11/AP_2020_1_Molnaretal.pdf
Megtekintés dátuma: 2021.02.20

Molnár Gyöngyvér – Pásztor-Kovács Anita (2015): A problémamegoldó gondolkodás mérése online tesztkörnyezetben. In: Csapó Benő – Zsolnai Anikó (szerk.): *Online Diagnosztikus mérések az iskola kezdő szakaszában.* Oktatáskutató és fejlesztő intézet, Budapest.
Elektronikus forrás: http://www.edu.u-szeged.hu/~csapo/publ/2015_OnlineDiagnosztikus.pdf
Megtekintés dátuma: 2021.02.17

NAT (2020): Nemzeti alaptanterv
Elektronikus forrás: https://net.jogtar.hu/jogszabaly?docid=a1200110.kor
Megtekintés dátuma: 2022.01.20.

NKP-DK5 (2020): Robotika, Algoritmizálás, Programozás. In: *Nemzeti közoktatási platform – Digitális kultúra 5. (NAT2020)*
Elektronikus forrás: https://www.nkp.hu/tankonyv/digitalis_kultura_5_nat2020/
Megtekintés dátuma: 2022.01.26

NKP-DK6 (2020): Robotika, Algoritmizálás, Programozás. In: *Nemzeti közoktatási platform – Digitális kultúra 6. (NAT2020)*
Elektronikus forrás: https://www.nkp.hu/tankonyv/digitalis_kultura_6_nat2020/
Megtekintés dátuma: 2022.01.26

NKP-DK9 (2020): Algoritmizálás és programozási nyelv használata In: *Nemzeti közoktatási platform – Digitális kultúra 9. (NAT2020)*
Elektronikus forrás: https://www.nkp.hu/tankonyv/digitalis_kultura_9_nat2020/lecke_05_001
Megtekintés dátuma: 2022.01.22

NKP-DK10 (2020): Algoritmizálás és programozási nyelv használata In: *Nemzeti közoktatási platform – Digitális kultúra 10. (NAT2020)*
Elektronikus forrás: https://www.nkp.hu/tankonyv/digitalis_kultura_10_nat2020/
Megtekintés dátuma: 2022.01.26

Orosz Sándor (1986): *Az oktatás mint a tanulás szabályozása*. Országos Oktatástechnikai Központ. Veszprém

Pajor Gabriella (2015): „Gyorsabban, magasabbra, bátrabban” – De hogyan? Teljesítménymotiváció iskolai környezetben. *Iskolapszichológia Füzetek* 34. sz.
Elektronikus forrás: https://www.eltereader.hu/media/2015/11/IP34_READER.pdf
Megtekintés dátuma: 2021.02.20

Pásztor Attila (2014): Innovatív eszközök rövid és hosszú távú hatásvizsgálata a programozás oktatásban. In: Torgyik Judit (szerk.): *Sokszínű pedagógiai kultúra.* ISBN 978-80-89691-05-0
Elektronikus forrás: http://www.irisro.org/pedagogia2014januar/0405PasztorAttila.pdf
Megtekintés dátuma: 2021.02.20

Pentelényi Pál (2011): Az algoritmikus szemléletmód kialakítása és fejlesztése. In: Tóth Péter (szerk.): *Kutatási Füzetek IV.* Óbudai Egyetem. Budapest

Perjés István – Vass Vilmos (2008): A curriculumelmélet műfaji fejlődése. *Új pedagógiai szemle 58. évf. 3. sz.*
Elektronikus forrás: https://ofi.oh.gov.hu/tudastar/perjes-istvan-vass
Megtekintés dátuma: 2022.01.21

Pléh Csaba (2013): *A megismerés tudománya – Az embertől a gépig és vissza*. Typotex. Budapest

Pólya György (1977): *A gondolkodás iskolája.* Gondolat kiadó. Budapest

Pólya György (2010): *A problémamegoldás iskolája I.* Typotex. Budapest.

PPT (2020): *Programterv Szoftverfejlesztő és -tesztelő szakmához*
Elektronikus forrás:
https://api.ikk.hu/storage/uploads/files/ptt_informatika_szoftverfejleszto_es_-tesztelo_2020pdf-1599123461507.pdf
Megtekintés dátuma: 2022.01.20

Réthy Endréné (2002): A kognitív és motivációs önszabályozást kialakító oktatás. *Iskolakultúra.* 2 sz.

Elektronikus forrás: http://epa.oszk.hu/00000/00011/00057/pdf/tanulm2002-2.pdf
Megtekintés dátuma: 2021.02.20

Richard C. Atkinson – Ernest Hilgard – Edward E. Smith – Susan Nolen-Hoeksema – Barbara L. Fredrickson – Geoffrey R. Loftus (2005): A pszichológiai fejlődés. In: *Pszichológia.* Osiris Kiadó

Robert C. Martin (2017): *Clean Architecture: A Craftsman's Guide to Software Structure and Design.* Pearson

Skemp, Richard R. (2005 [1975]): *A matematikatanulás pszichológiája*. Edge 2000 kiadó. Budapest.

Szlávi Péter – Törley Gábor – Zsakó László (2019): *Programming Theorems Have the Same Origin*
Elektronikus forrás:
https://www.researchgate.net/publication/336095439_Programming_Theorems_Have_the_Same_Origin
Megtekintés dátuma: 2022.02.22

Szlávi Péter – Zsakó László (2016): *A programozás gondolkodási eszköztára – Algoritmikus absztrakció, dekompozíció-szuperpozíció.*
Elektronikus forrás: (PDF) A programozás gondolkodási eszköztára – Algoritmikus absztrakció, dekompozíció-szuperpozíció (researchgate.net)
Megtekintés dátuma: 2021.02.17

Tasnádi Péter (2017): Mit várunk a hallgatóktól? Absztrakció, megértés, alkalmazás. In: Károly Krisztina, Homonnay Zoltán (szerk.): *Diszciplínák tanítása – a tanítás diszciplínái 4. – A tanulás és a tanítás értékelése.* ELTE Eötvös Kiadó. Budapest.
Elektronikus forrás: http://www.eltereader.hu/media/2017/07/Diszciplinak_4_READER.pdf
Megtekintés dátuma: 2021.02.17

Varga Tamás (2001): *Matematika lexikon – matematikatanároknak, szülőknek, matematikát tanulóknak.* Műszaki könyvkiadó. SHL Hungary Kft. Budapest

Winkler Márta (2015): *Kinek kaloda, kinek fészek.* Edge 2000 kiadó

Woodkock Jon (2017): *Star Wars Coding Projects: A Step-by-Step Visual Guide to Coding Your Own Animations, Games, Simulations and More!* Dk Children

Whitney David (2021): Kódolj! – *Tanulj meg programozni HTML-ben, CSS-ben és JAVAScriptben!* Scolar Kiadó Kft.

Érettségi feladatsorok

2018tk: Informatika középszintű gyakorlati vizsga, 2018. május 17.
Elektronikus forrás: https://dload-oktatas.educatio.hu/erettsegi/feladatok_2018tavasz_kozep/k_inf_18maj_fl.pdf
Megtekintés dátuma: 2021.02.21.

2020te: Informatikai alapismeretek emelt szintű gyakorlati vizsga
Elektronikus forrás: https://dload-oktatas.educatio.hu/erettsegi/feladatok_2020tavasz_emelt/e_infoism_20maj_fl.pdf
Megtekintés dátuma: 2021.02.21

2019ok: Informatika ismeretek középszintű gyakorlati vizsga
Elektronikus forrás: https://dload-oktatas.educatio.hu/erettsegi/feladatok_2019osz_kozep/k_infoism_19okt_fl.pdf
Megtekintés dátuma: 2021.02.21

Ábrajegyzék