„ProgAlap” és ami mögötte van

Szalayné Tahy Zsuzsanna[1], Czirkos Zoltán[2]

[1] sztzs@caesar.elte.hu
ELTE IK
[2] czirkos@eet.bme.hu
BME VIK

Absztrakt. Az egyetemi programozás-oktatás két legjelentősebb hazai központja az ELTE IK illetve a BME-VIK. A BME-s „A programozás alapjai”, illetve az ELTE-s „Programozási alapismeretek” tárgyak összehasonlító elemzése során bemutatjuk, a képzések célját, az egyes tantervek súlypontjait, a közös és eltérő vonásokat. A két képzés esetén néhány konkrét feladaton keresztül bemutatjuk a hallgatóktól elvárt előismereteket, képességeket és készségeket, a feladatnak a képzésben betöltött szerepét, a hallgatótól elvárt fejlődést. Vizsgáljuk az elsajátítást akadályozó tényezőket, a hallgatók lemaradásának okait. A tapasztalatok alapján javaslatot teszünk a ProgAlap-okat megelőző oktatás tartalmára.

Bevezetés

Code Week [1], Hour of Code [2], <Code_4_Education> [3], World Robot Olympiad [4]… A világ számos országában fókuszba került a kódolás, a programozás [5] [6] [7]. Elterjedőben van a nézet, miszerint a felnövekvő nemzedék minden tagjának meg kell tanulnia kódolni, programozni [8] [9] [10] [11]. Egyik oldalról, a munkaerőpiacon nagyon erős igény van programozó szakemberekre. Másik oldalon egyre szignifikánsabb igény, hogy a felhasználó – az átlagember – értse környezetének működését, aktív, alakító résztvevője legyen a társadalomnak, aminek egyik feltétele, hogy ismerje a digitális világot létrehozó és alakító elemét, a programozást.

A programozás összetett tevékenység, számos ismeret, képesség és készség meglétét jelenti, amelyek elsajátítása, fejlesztése különböző módszerekkel lehetséges. Az egyes módszerek jellemzője ezen ismeretek és készségek tanításának sorrendje illetve hangsúlyozása [12].

- eljárás-, algoritmus-orientált
- adatorientált
- specifikáció orientált
- problématípus-orientált
- nyelvorientált
- utasításorientált
- matematikaorientált
- hardverorientált
- modell-, mintaorientált.

A programozás oktatásának számos célja lehet. A felsőoktatásban e célok egyikét előtérbe helyezve oktatják a „programozás alapjait”. Az oktatás módszerét jellemzően a cél határozza meg, és jellemző, hogy elvileg „nulláról indulva” tanítja a programozást.

Két egyetemi tantárgy oktatását elemezzük: a BME-VIK mérnökinformatikus képzés első félévben oktatott „A programozás alapjai” tárgyat (BME-ProgAlap) [13], illetve az ELTE IK programtervező matematikus képzésében első félévben oktatott „Programozási alapismeretek” tantárgyát (ELTE-ProgAlap) [14].

A BME-ProgAlap képzése nem csak mérnökinformatikusoknak szól, a villamosmérnökök is lényegében azonos tematika mentén tanulják a ProgAlap-ot, míg az ELTE-ProgAlap képzése in-

formatikatanárok számára is hasonló. A BME-ProgAlap a mérnöki, műszaki szemléletnek megfelelően a nyelv- illetve hardverorientált módszert részesíti előnyben, az ELTE-ProgAlap a matematika oldaláról közelíti a témát, így a specifikáció- illetve az algoritmusorientált módszert preferálja. A képzés során sokszor ugyanazt a feladatot, problémát tárgyalják, de – szemléletmódjukból eredően – mind az elvárt, mind a megtanított ismeretek illetve készségek terén nagy eltérést tapasztalunk. Az elvárt ismeretek megjelenése felveti a kérdést, hogy mit jelent a „nulláról indulás". Az elvárt ismereteken keresztül a tantárgyi követelményeket, illetve a követelmények teljesítésének sikerességét is vizsgáljuk. A kutatás során szeretnénk meghatározni azt a „nulla" szintet, ami a két képzés sikeres teljesítéséhez elvárt, amit a középiskolai oktatás során a diákoknak meg kellene tanulniuk.

Célkitűzések, tematika

A tantárgyak oktatási céljairól a tantárgyi adatlapok alapján nyerhetünk képet.

A BME-ProgAlap célja:

> *A tantárgy célkitűzése, hogy a hallgatók készség szinten alkalmazható ismereteket szerezzenek a számítógépes problémamegoldás módszereinek és alapvető eszközeinek használatában. További cél, hogy a megszerzett ismereteket és készségeket további tanulmányaik során hatékonyan legyenek képesek alkalmazni.*
>
> *A célkitűzés teljesítését egy magas szintű programozási nyelv, a C megismerése teszi lehetővé. A gyakorlatok anyaga folyamatosan követi az előadások tematikáját, azok mélyebb megértését, az algoritmusok részletes megismerését támogatja. A laboratórium célja, hogy a hallgatók gyakorlati jártasságot szerezzenek az előadáson és gyakorlaton megismert módszerek kipróbálása által, és készség szinten elsajátítsák a fejlesztőkörnyezetek használatát.*
>
> *Az anyag jobb elsajátítása érdekében a laboratóriumi foglalkozásokhoz kapcsolódóan egy nagyobb házi feladatot is meg kell oldani. [15]*

Az ELTE-ProgAlap célja:

> *Probléma megoldási stratégiák, az informatikai problémamegoldás alapjai. A problémák megoldásához szükséges informatikai eszközök és módszerek. A problémamegoldás lépései. Mi a programozás, a programkészítés folyamata. A feladat és a program. Programkészítési elvek. Algoritmikus struktúrák és adattípusok kialakulása - programozási modellek. Algoritmus-leíró eszközök.*
>
> *Konstans, változó, típus fogalma. Elemi és összetett adatok, file-szervezés. Egész és valós számok, logikai értékek, karakterek. Tömbök, szövegek, rekordok - ezek fogalma, használata.*
>
> *Algoritmikus struktúrák. Elemi algoritmusok típusfeladatokra. Összegzés, eldöntés, keresés, számlálás, maximum-kiválasztás, kiválogatás, rendezések.*

Egy programozási nyelv alapvető elemei (értékadás, beolvasás, kiírás, elágazások, ciklusok, eljárások, függvények, konstansok, változók, típusok definiálása). A programfejlesztői környezet, szerkesztés, fordítás, futtatás.

A programkészítés, mint termék előállítási folyamat (feladat-meghatározás, tervezés, kódolás, tesztelés, hibakeresés, hatékonyság- és minőségvizsgálat, dokumentálás) elemi szinten.

Nyelvi, tervezési, kódolási, kipróbálási eszközök, amelyek az algoritmusok, illetve a kitűzött feladatok megvalósításához szükségesek. Egy programozási nyelv (C++) elemeinek megismerése, alkalmazása. Alapvető tesztelési, hibakeresési módszerek, ezek támogatása egy programozási nyelv fejlesztői környezetében. Az elkészült programok minőségi elemzése. [16]

Táblázatosan áttekintve a főbb szempontokat észrevehetjük, hogy az apró eltérések mögött a **programozás** is, **mint tevékenység, eltérő értelmezésű a két képzésben**.

	BME-ProgAlap	ELTE-ProgAlap
Problémamegoldás	készség szintű eszközhasználat	eszközfüggetlen stratégia
Algoritmus	megismerés	elemzés
Nyelv	C nyelv elsajátítása	C++ eleminek megismerése
Adatstruktúrák [17]	gyakorlati, implementáció (int*)	elméleti, matematikai definíció (ℕ)
Programkészítés	gyakorlata	folyamata
Tesztelés	debug	tesztadatok készítése
Feladatmegoldás	részekre bontás, adatok definiálása, részek implementálása	specifikáció, algoritmus, esetenként kódolás

1. Célok

Az előadás jegyzetekben, útmutatókban is látható a különbség. Néhány szempontot kiemelve megfigyelhetjük, az elvárt hallgatói tevékenységek közötti különbségeket. :

BME-ProgAlap	ELTE-ProgAlap
„feladatmegoldás” tevékenység	
Minden feladathoz megfontolások, megoldandó problémák felsorolása, kódrészletek, a kód egyes részeinek magyarázata.	Minden feladathoz specifikáció, matematika nyelvén megfogalmazva. Ebből származtatva a struktogram, néha a megvalósítás teljes (kommentekkel együtt) kódja.
„problémamegoldás” tevékenység	
Adott probléma megoldását a megoldás helyes kódolása jelenti.	Adott probléma megoldása az absztrakció elméleti származtatását jelenti, fő feladat a probléma vizsgálata, algoritmizálás.
input ellenőrzése	

A beolvasott adatok ellenőrzése egy feladat, amit a dolgozatban nem kell leírni.	A beolvasott adatok ellenőrzése minden esetben.
programvezérlési eszközök használata	
while() és for() ciklusok szervezése praktikusan. for() ciklusban a ciklusfeltétel lehet összetett, for() javasolt pointerléptetéssel listához végjeles beolvasáshoz is	while() és for() ciklusszervezés tudatos, az algoritmus struktúrájából következő választás. for() ciklus csak számlálós ismétlésre használható, más vagy több feltétel esetén while() ciklus használandó.
goto, break, continue alkalmazása megfontolandó, általában van szebb megoldás is.	goto, break, continue (ami a struktogramba nem fér bele) alkalmazása tilos

2. Néhány gyakorlati különbség

A célok megfogalmazásából látható, hogy az ELTE-ProgAlap a nagy programok tervezésére, algoritmizálásra helyezi a hangsúlyt, míg a BME-ProgAlap az eszköz-közeli gyakorlati problémák megoldását tartja fontosnak. Az ELTE-ProgAlap a programozást egyfajta matematikai leírónyelvként tanítja, amelyben a kódolás a gondolatsor, az elméleti megoldás ellenőrzése. A BME-ProgAlap a programozást gyakorlati tevékenységnek tekinti, a kódolás a számítógéppel történő kommunikáció eszköze. Természetesen szükség van ehhez elméleti tudásra, meg kell tanulni a számítógép működésének logikáját és e logika mentén kell megfogalmazni az utasításokat.

A ProgAlap oktatásának gyakorlata

A célok és tematikák a ProgAlap tantárgyak hivatalos meghatározásai. A két képzés közötti eltéréseket úgy érthetjük meg igazán, ha közös témák tanításának gyakorlatát vizsgáljuk. Mindkét képzésre jellemző, hogy az előadásokhoz készített prezentáció elérhető, egyben jegyzetként is használható vázlat. A prezentációk azonban csak az élő előadással együtt nyernek igazán értelmet. A következőkben bemutatunk egy-egy jegyzetrészletet, a hozzájuk kapcsolódó, szóban elhangzó kiegészítéseket, illetve azokat a megjegyzéseket, szempontokat, amelyek a másik képzés szemléletéből adódnak.

Eldöntés tételének alkalmazása a BME-ProgAlap-n

Az Eldöntés tételét a 3. előadáson tanulják a hallgatók, mondatszerű leírással és néhány példával [17]. A 7. héten újra előkerül a téma, a tömbök algoritmusainak tanítása során. A feladat a VAN-E() függvény megírása. A keretezett részek az előadásjegyzetből származnak [18], e szószerinti idézeteket követik az adott részlethez hozzáfűzött megjegyzések.

A következő példák double[]-ben keresnek adott elemet

```
int van_e(double *szamok, int meret, double keresett) {
   int i, van_talalat;
   van_talalat = 0;
   i = 0;
   while (i < meret && !van_talalat) {
      if (szamok[i] == keresett)
         van_talalat = 1;
      ++i;
   }
```

```
    return van_talalat; // miért állt meg a ciklus?
}
```

„Van-e?” – elég egy olyat találni, és megállhat a ciklus

Ha egyet sem találtunk, akkor hamis érték marad a változóban

A van-e függvényt egyéb formákban is lekódolhatjuk. Ezek mind ugyanolyan jól működnek, mint a fenti változat.

BME-ProgAlap: Ez a megoldás a 3. előadáson tanított mondatszerű leírás C-ben, függvényként kódolt megfelelője. Sok elhagyható dolog van benne, ami kódoláskor hibalehetőséget jelent, nehezen átlátható. További problémája a megoldási módszer tanításának, hogy a mondatszerű leírásban az eredmény logikai típusú, ami hiányzik a C nyelvből.

ELTE-ProgAlap: Ez a megoldás megfelel az ELTE-n struktogrammal megadott megoldásnak, illetve annak, ami a BME-ProgAlap 3. előadásában található.

for() összetett feltétellel

```
van_talalat = 0;
for (i = 0; i < meret && !van_talalat; ++i)
   if (szamok[i] == keresett)
      van_talalat = 1;
return van_talalat;
```

A C bármilyen kifejezést enged a for() ciklus fejlécében, így ott egy összetett kifejezés is lehet.

BME-ProgAlap: A C nyelv és általában a ma használatos nyelvek engedik a ciklusfejben több feltétel megadását. A kód így sokkal biztonságosabban írható, a for() ciklusfej átláthatóbb, rendezettebb, mint a while(); könnyebben észrevehető, ha hibás. Például szinte soha nem fordul elő, hogy lemaradna az iterátor növelése.

ELTE-ProgAlap: A strukturált programozás nem engedi, hogy a számlálós ciklusban több feltétel szerepeljen, a számlálós ciklus az elöl tesztelő ciklus egy speciális esete, ahol a feltétel egy iterátor vizsgálata. Azok a hallgatók, akik követik a programozás lépéseinek tanult sorrendjét, ilyen kódot nem írnak, mert a struktogramból egyértelműen következik a while() ciklus használata. Nem jó, ha a hallgató megfordítja a programkészítés sorrendjét, – azaz először kódol, majd ezt követően találja ki a struktogramot és a specifikációt – mert ez a gyakorlat később komoly problémákat eredményez.

A keresést megszakítva

```
for (i = 0; i < meret; ++i)
   if (szamok[i] == keresett)
      return 1;
return 0;
```

Mivel a C-ben szabad a függvény belsejében is visszatérni, egy találat esetén azonnal visszatérünk igaz válasszal a „van-e” kérdésre.

```
for (i = 0; i < meret; ++i)
   if (szamok[i] == keresett)
      break;
return i < meret;
```

Így is lehetne. Megszakítjuk a kereső ciklust, ha megtaláltuk az elemet. Utána kell egy feltétel, ami igazra értékelődik ki, ha megtaláltuk az elemet – ehhez figyelni kell azt, hogy az i<meret miatt állt meg.

BME-ProgAlap: Ez a megoldás átláthatóbb, a természetes gondolkodást követi: „végigmegyek az elemeken, ha megtalálom, amit keresek, akkor egy nem nulla értékkel térek vissza, egyébként a hamisnak megfelelő 0 lesz az eredmény.” Ismét felmerül, hogy érthetőbb lenne a megoldás, ha a C nyelvben lenne logikai típus, de ez kevés hallgatónál jelent problémát.

ELTE-ProgAlap: Struktogramban, modellekben minden függvény egy ponton, a végén tér vissza. Ez a megoldás elfogadhatatlan, a modellből nem származtatható. A második kódrészlet jól mutatja a probléma gyökerét, a „break;” nem modellezhető.

Lineáris keresés: hol van?

```
int hol_van(double *szamok, int meret, double mit) {
   int i;
   for (i = 0; i < meret; ++i)
      if (szamok[i] == mit)
         return i;
   return -1;
}
```

A nincs találatra a –1-et és a tömb méretét is elterjedten használják. Mindkettő a tömb indexeinek tartományán kívül esik (mivel az 0…meret–1), ezért mindkettő egyformán jó lehet. A –1 előnye, hogy szembetűnőbb; a méret előnye, hogy nem negatív szám (mivel a tömb mérete amúgy sem lehet negatív, így elvileg a méreteket, indexeket tárolhatnánk előjel nélküli változókban is).

BME-ProgAlap: Az eldöntés tételének alkalmazása a lineáris keresésen belül, csak abban van eltérés, hogy a visszaadott érték nem logikai, hanem a találat helye.

ELTE-ProgAlap: A megoldás nem származtatható a keresés tételének struktogramjából. A függvényben kétszer szerepel a „return”, ami nem írható le a modellben, a kódolás nem követi a tervezést. A megoldás nem strukturált, csúnya.

Az oktatott tananyag az eldöntés illetve a lineáris keresés függvényként implementálása, azonban indirekt módon programozási, kódolási technikát is tanul a hallgató. Fontos azt is látni, hogy a megoldási módokkal gondolkodási módszereket is tanítunk. A két képzésben e tananyagrész során talán ebben van a legnagyobb eltérés.

Az ELTE-ProgAlap a programtervezés lépéseit helyezi előtérbe:

1. Specifikáció. Az igazi a matematikai leírás lenne, de első félévben elegendő annak felismerése, hogy eldöntés, keresés… típusú feladatról van szó, melyek az input adatok, erre milyen feltételeknek kell teljesülnie, milyen típusú lesz az output, illetve a megoldás melyik tétel alkalmazásának felel meg.
2. Algoritmizálás: Az ELTE-ProgAlap általánosan elvárt formában, struktogrammal adja meg a megoldást.
3. Kódolás: Egy megadott nyelven – jelenleg C++-ban – a struktogramot lekódoljuk.
4. …

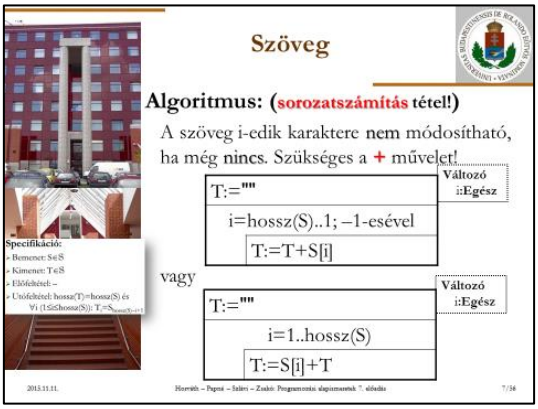
A BME-ProgAlap az elemi feladatok megoldásának ésszerű, logikus kóddá alakítását tartja fontosnak. Arra helyezi a hangsúlyt, hogy a kód a többi programozó számára is könnyen értelmezhető, informatív legyen, a compiler (beépített eszköz nélkül is) optimális kódot tudjon készíteni.

Sorozatszámítás tételének alkalmazása az ELTE-ProgAlap-n

A Sorozatszámítás tétele a 4. előadás témája [19]. A 7. előadás a tanult tételek alkalmazásáról szól, azt vizsgálja, hogy egyes feladatok specifikációja során hogyan lehet eldönteni, hogy melyik programozási tételek használhatók a megoldáshoz. Ezek közül az egyik feladat egy szó (szöveg) visszafelé írása. Az előadáson az itt bemutatott két dia kíséri a feladat megoldását, [20], az egeys diákat a hozzáfűzött megjegyzések követik.

ELTE-ProgAlap: Az első lépés a feladat specifikálása. Ennek során felmerül, hogy a feladat megoldása másolás, de ez a sorrendfordítás miatt nem igaz. A feladat egy példa az utófeltétel megfogalmazására. Az utófeltétel pontos megadása nem követelmény, de a kifejezést olvasni, azaz értelmezni, tudni kell. A feladat matematikai megfogalmazása a hallgatók számára nehéz, ezért az első félévben minden lehetséges helyzetben bemutatásra kerül, bár reprodukálása nem követelmény.

BME-ProgAlap: Ennek a feladatnak rengeteg megoldása létezik. Nem kötődik programozási tételhez, de fontos, hogy a hallgató meg tudja fogalmazni a megoldás menetét a vezérlési szerkezetek felhasználásával. Az itt látható specifikáció helyett fontos az éppen tanított témának megfelelő adatstruktúra illetve kódolási eszközök megadása. Például függvényt vagy kódrészletet kér; előre definiált változó vagy memóriafoglalás használatát várja el a feladat. Esetenként az eredeti változóban kell megjelennie a végeredménynek.



ELTE-ProgAlap: A feladat valójában sorozatszámítás tételének felismerésére ad példát. A struktogramból látszik, hogy a megoldás a sorozatszámítás struktúráját követi, de eltéréseket

is tartalmaz. Az első struktogram ciklusában a számlálás csökkenő. A második megoldás szebben mutatja, a sorozatszámítás tételének alkalmazását. Az egyetlen „furcsaság”, hogy a kumulálás T ≔ T + S[i] helyett T ≔ S[i] + T. Fel kell hívni a hallgatók figyelmét arra, hogy a konkatenáció nem kommutatív.

BME-ProgAlap: Bár a második megoldásban a ciklus a megszokott, de eközben egy O(N) algoritmusból $O(N^2)$ lett. Az első megoldás még kódolható T += S[i] formában, ahol megoldható lesz, hogy ne kelljen a stringnek minden karakternél átméreteződnie (nagyobb tömb foglalásával). A második megoldásnál viszont minden egyes összefűzés egy másolással fog járni. A gyakorlattól teljesen idegen, hogy a string 1-től indexelődjön, ezért már tervezéskor 0-tól indexeljük. Gyakorlatokon a szöveg helyben megfordítása is szerepel. Inkább a csere eljárás megírásának gyakorlása teszi aktuálissá a feladatot, nem a tételek felismerése. Például:

```
forditott = eredeti;
for (i = 0; i < forditott.size()/2; ++i)
   swap(forditott[i], forditott[forditott.size()-1-i]);
```

Ez a megoldás azért is szép, mert a forditott = eredeti; pillanatban épp a megfelelő méretű string jön létre.

A feladathoz fűzött megjegyzésekből jól látszik, hogy a két képzés más gondolkodási sémát tanít a feladatok megoldására. A BME-ProgAlap a megvalósítás eszközeit veszi alapnak, erre építi a megoldását. A gondolkodást vezérlő kérdés: Milyen adatstruktúrákat használjunk, hogyan fogja elvégezni a számítógép az utasításainkat? Az ELTE-ProgAlap a modellezési sémákat keresi a feladatban. Kérdése: Melyik ismert algoritmus – programozási tétel – alkalmazható a megoldásra? Gondolatmenetének alapja, hogy az ismert algoritmus helyessége bizonyított, kódolása automatizálható. Jellemző különbség a kétféle gondolkodásmód között, hogy a BME-ProgAlap minden esetben beszédes elnevezéseket használ, míg az ELTE-ProgAlap – épp az általánosíthatóság kiemelése miatt, a matematikai hagyományokat követve – a specifikáció és struktogram szintjén előszeretettel használ egykarakteres jelöléseket.

Kihatások

A két példa bemutatja a két képzés tanítási gyakorlatában szinte minden pillanatban megjelenő eltérő szempontokat, nézeteket. Ezek az eltérő szempontok és nézetek alapvetően befolyásolják a hallgatók gondolkodását. Az egyik képzésen képviselt megoldásokat a másik képzésen értékelhetetlennek minősítik (ELTE-ProgAlapon 0 értékű zárthelyi, ha nem az előadáson tanult séma algoritmus szerinti, BME-ProgAlapon „halálfejes hiba”, azaz a pontszám felének elvesztését jelenti az értelmetlen indexelés.). Az ellentétek az egyetemek falain kívül is megjelennek: Az ELTE-n végzett informatikatanárok értetlenül állnak azon diákok „lázadásával” szemben, akik a BME-ProgAlap-nak megfelelő végzettségű ismerősöktől tanulják, lesik el a programozás alapjait.

A ProgAlap-ok problémái

Láthatjuk, hogy a két képzés céljában és tartalmában épp csak sejthető különbségek mögött jelentős gondolkodásmódbeli különbség is van. Mindkét képzésre igaz, hogy a másik képzés gondolkodásmódját későbbi tantárgyakban tanulják a hallgatók, azaz elvileg az MSc képzés első éve után, már hasonló lesz a hallgatók tudása. Persze csak azoké, akik eljutnak odáig. Jellemző, hogy csak a BSc-t végzik el a hallgatók, de az igazi probléma mindkét képzés esetén, a nagymértékű korai lemorzsolódás. A lemorzsolódásnak számos oka lehet, amelyek közül most csak az előismeret illetve az oktatás módszereire fókuszálunk.

Mitől függ az eredményesség BME-ProgAlapon?

A BME-n egy 571 fős évfolyamot vizsgálunk, akik közül 246 hallgató töltötte ki tanév elején a kérdőívet. Az első kis zárthelyi, az első négy kis zárthelyi összesített eredménye és az első nagy zárthelyi eredményét vetettük össze a kérdőíves felmérés válaszaival (ezek a grafikonokon rendre kZH1, kZHk és NZH jelzéssel szerepelnek). A kérdőíves felmérésben megkértük a hallgatókat, hogy adják meg, milyen tudással, képességekkel érkeztek – részben valós eredményeket kértünk, részben becslést saját tudására –. A válaszokat osztályzatokkal jellemeztük. Megvizsgáltuk, hogy hogyan korrelál a zárthelyik pontszáma a kérdőíven kérdezett adatokkal. A kérdőíves felmérés és a dolgozatok eredménye közötti kapcsolat az alábbi kérdéseknél a legerősebb (NZH>0,25):

4. Milyen adatstruktúrákat használtál?

Semmilyet (nem tudom, hogy mit jelent az adatstruktúra)	1
Egyszerű változókat	2
Változókat és tömböket	3
Változókat, tömböket és listákat	4
Változókat, tömböket, struktúrát (rekordot)	5
Változókat, tömböket, osztályokat	5
Változókat, tömböket, listákat, struktúrákat, osztályokat. Ezekből legalább 4-et.	5

6. Emelt szintű érettségi programozás feladatán hány pontot értél el | hány pontot tudtál volna elérni?

0 – 3 (semennyit)	1
4 – 10 (pl. beolvasás, 1-2 feladat)	2
11 – 20 (pl. beolvasás, könnyű feladatok)	3
21 – 30 (kb. a fele)	4
31 – 40 (lényegében meg tudom oldani az ilyen típusú feladatokat)	5
41 – 45 (kb. maxpontot, néha benézek 1-2 részletet)	5
nem tudom megítélni, nem láttam még ilyen feladatsort.	1

11. Milyen osztályzatot szereztél matematika érettségin?

2 (közép vagy emelt szinten)	2
3 (közép vagy emelt szinten)	3
4 (közép vagy emelt szinten)	4
5 középszinten, de emelt szinten nem tudtam volna 5-ösre	5
5 emelt szinten vagy középszinten, de emelt szinten is meglett volna, csak nem volt rá szükségem	5

12. Hányasra értékeled fizikatudásodat?

1 (az utóbbi néhány évben nem tanultam fizikát, de korábban sem igazán)	1
2 (az s=v*t azért még megy)	2
3 (alapvető összefüggéseket ismerem)	3
4 (függvénytáblázattal az egyszerű feladatokat meg tudom oldani)	4
5 (értem a fizikát, függvénytáblázattal összetett, több képletet igénylő feladatokat is meg tudok oldani)	5

21. A Code::Blocks fejlesztő környezetet félévkezdést megelőzően...
    - ... is ismertem, használtam programozáshoz 5
    - ... is ismertem, de nem használtam 4
    - ... nem ismertem, de hasonló struktúrájú programozási környezetet használtam 4
    - ... nem ismertem, de hasonló struktúrájú alkalmazást (pl. grafikus szoftver) már használtam 3
    - ... nem ismertem és hasonlót sem használtam. 1

25. A számok bináris ábrázolásáról a szemeszter kezdete előtt...
    - ... nem tanultam. 1
    - ... tanultam, ismertem a kettes számrendszert, tudtam oda-vissza váltani 2-es és 10-es számrendszer között 4
    - ... tanultam, ismertem a kettes számrendszert, átváltásokat és az összeadást is. 5
    - ... tanultam, az átváltást, a műveletek végzését is tanultam és negatív számokra az abszolútértékes előjeles és kettes komplemens módú ábrázolást is. 5

28. Programozási tételeket korábban...
    - ... nem tanultam, teljesen új volt számomra előadáson 2
    - ... nem tanultam, de már írtam hasonló jellegű kódot. 4
    - ... tanultam, de soha nem értettem, hogy minek. 2
    - ... tanultam, szerintem az érettségin is lényegében ezeket az algoritmusokat kell tudni. 5

kID	kZH1	kZHk	NZH	diagram	Kérdés
Átlagpont	0,28	0,37	0,45		
VálaszDB	0,17	0,29	0,32		
4	0,31	0,30	0,37		Milyen adatstruktúrákat használtál?
6	0,27	0,30	0,37		Emelt szintű érettségi programozás feladatán hány pontot értél el / hány pontot tudtál volna elérni?
11	0,03	0,22	0,30		Milyen osztályzatot szereztél matematika érettségin?
12	0,09	0,16	0,29		Hányasra értékeled fizikatudásodat?
21	0,26	0,30	0,38		A Code::Blocks fejlesztő környezetet félévkezdést megelőzően...
25	0,10	0,18	0,25		A számok bináris ábrázolásáról a szemeszter kezdete előtt...
28	0,26	0,28	0,28		Programozási tételeket korábban...

A dolgozateredményekkel leginkább korreláló hozott ismeretek

A válaszokból kitűnik, hogy az első néhány hét során egyre fontosabbá válik a hozott matematika- és fizikatudás minősége. Ez három tényezőnek tudható be: 1) a feladatok egyre jobban igénylik ezen témakörök ismeretét; 2) a szakon tanult további tárgyak sikeressége – a szükséges időráfordítás illetve jövőbeli kilátásokon keresztül – kihat a ProgAlap eredményére is; 3) hasonló absztrakciós készségeket gondolkodási sémákat igényelnek.

A tárgy teljesítése szempontjából meghatározó, hogy korábban hallott-e a hallgató adatstruktúrákról programozási tételekről, számok bináris ábrázolásáról; van-e gyakorlata a képzésben

használt programozási környezetben való munkára. Összességében azonban, a legerősebb kapcsolat az emelt szintű érettségiek megfelelő tudással van.

Több kérdés vonatkozott az informatika egyes területeire, nyelvismeretre, nyelvtanulási képességre, illetve a gépírástudásra. Az ezekkel együtt számolt átlag kiemelkedően magas korrelációt mutat a zárthelyik eredményével, ami azt mutatja, hogy a teljesítést a hozott tudás és képességek széles spektruma határozza meg. **Nincs olyan tudáselem, ami elegendő, vagy aminek hiánya végzetes, a képességek és ismeretek együtthatása jelentősebben befolyásolja a végeredményt, mint egyenként.**

A másik speciális szempont a kérdésekre adott válaszok számának és az eredmények összehasonlítása. Ez mutatja a tárgyhoz, tanuláshoz való hozzáállás függését az eredményekhez. A kérdések az 1. illetve 3. héten, a tárgy honlapján jelentek meg, a válaszadás önkéntes volt. A lelkesebbek, a feladatokat precízebben teljesítők töltötték ki a kérdőívet. Úgy tűnik, ez a hozzáállás segít a jobb eredmények elérésében.

A vizsgálat másik irányban, a legkevésbé releváns ismeretek körében is hozott érdekes eredményeket.

14. Általában milyen jegyeket kaptál helyesírásra?

Nem értékelték	1
Gyakran volt elégtelen	2
2 vagy 3 (rossz helyesíró vagyok)	3
4 vagy 5 (jó vagyok helyesírásból	5

17. Gépírás 1: Melyik igaz leginkább?

Tudok 10 ujjal vakon gépírni.	5
Nem 10, de legalább 5 ujjal tudok gépírni.	4
2-4 ujjal tudok gépelni.	3
Eddig nem szereztem gyakorlatot gépírásból, többnyire képernyő-billentyűzetet használtam, vagy azt sem.	2

18. Gépírás 2: Melyik a legjellemzőbb?

Tudok beszélgetés közben levelet írni.	5
Tudok lapról szöveget begépelni úgy, hogy közben csak a lapot nézem.	5
Gondolataimat gondolkodás tempójában le tudom gépelni.	5
Gyorsabban írok kézzel, mint géppel.	4
Fárasztó a betűk keresgélése.	3
Egyik sem jellemző	2

22. A Code::Blocks...

... telepítése és használata nem okozott gondot.	5
... a telepítésével voltak problémáim, de használata könnyű.	4
... telepítése könnyen ment, de nagyon nehézkes a használata, sokat kell(ene) keresgélnem, hogy a szükséges menüpontokat, funkciókat megtaláljam	3
... telepítésével és használatával sok problémám van.	1

23. Angol nyelvtudásom az egyetemen használt programok, források megértéséhez...

... bőven elegendő	5

... nagyjából jó, néha utána kell nézni egy-egy kifejezésnek 4
... nagyjából jó, néha a szövegkörnyezetből találom ki, hogy miről van szó. 3
... nem az igazi. Segítség kell a szöveg (pl. hibaüzenetek) megértéséhez 2
... nem elég. Nem értem, ami angolul van, magyarul keresek segítséget. 1

24. A struktogram...
... a vezérlési szerkezetek szemléletes ábrázolása 5
... lehet, hogy van, akinek szemléletes, de inkább írjuk le. 4
... kimondani nem tudnám, de már láttam ilyen dobozokat 2
... hm, életemben nem hallottam ezt a szót. 1

27. Az implikációról a szemeszter kezdete előtt...
... nem tanultam. 2
... nem tanultam, most sem tudom, hogy mit jelent. 1
... tanultam, matematikából, pl. az indirekt bizonyítással kapcsolatban 4
... tanultam, matematikából és informatikából is, az implikációt fel tudtam írni más logikai műveletekkel. 5

kID	kZH1	kZHk	NZH	diagram	Kérdés
14	0,07	0,11	0,02		Általában milyen jegyeket kaptál helyesírásra?
17	-0,04	-0,06	-0,02		Gépírás 1: Melyik igaz leginkább?
18	-0,05	-0,11	-0,09		Gépírás 2: Melyik a legjellemzőbb?
22	0,03	0,06	0,05		A Code::Blocks...
23	0,06	0,04	0,10		Angol nyelvtudásom az egyetemen használt programok, források megértéséhez...
24	-0,03	0,05	0,08		A struktogram...
27	0,01	0,12	0,10		Az implikációról a szemeszter kezdete előtt...

A dolgozateredményekkel legkevésbé korreláló hozott ismeretek

Az eredmények azt mutatják, hogy a helyesírásnak (pl. betű kihagyások) a zárthelyik eredményére nincs jelentős hatása, valamint a gépírási képességek, technikák sem számítanak. A vizsgált zárthelyiket a hallgatók papíron írták, az értékelésnél a helyesírási problémákat igyekeztek tolerálni a javítók. Például kézírásban nem hiba a „\” és a „/” felcserélése. Az eredmény egybeesik azzal a közvélekedéssel, hogy a rossz helyesíróknak jó választás a programozás, mert a számítógép (és az értékelési elvek) kompenzálja a helyesírási nehézségeket. A gépelés sebességének problémája a vizsgált helyzetekben – kézírásos dolgozatok – nem számít.

A Code::Blocks telepítési nehézsége nem hatot ki az eredményességre. Tekintettel arra, hogy a fejlesztő környezet előzetes ismerete – az előző táblázat alapján számít, azt mondhatjuk, hogy a fejlesztőkörnyezet váltás nem jelent problémát, de az igen, ha a hallgató nem használt korábban ilyen típusú programot.

A struktogramot és az implikációt a BME-ProgAlapon lényegében nem tanulják (nem szerepel zárthelyiken), így ezen ismeretek megléte vagy hiánya a zárthelyikben nem derül ki. Az angol nyelvismeret is csak később válik fontossá, az első félévben nem releváns.

Mitől függ az eredményesség ELTE-ProgAlapon?

Az ELTE-ProgAlapon 16 hallgató fejlődését tudtuk figyelni. Az első gyakorlaton végzett felmérésen a BME-s hallgatókkal azonos kérdőívet töltötték ki, de a 3. heti felmérést már nem végeztük el, mert egyértelműen látszottak a következtetések. A csoport szinte minden tagja „túlkoros”, az átlagéletkor 27 év. Két csoportra oszthatók, azokra, akik még nem tanultak semmilyen nyelven programozni, illetve azokra, akik már valamikor, valamilyen nyelven programoznak. Az előbbi csoportnak komoly nehézségei vannak a képzés teljesítésével. Az utóbbiaknak legfeljebb elvi problémái vannak: a gyakorlati életben tapasztalt elvárások nem egyeznek a tárgy követelményeivel. Náluk lényegében a fentebb bemutatott gondolkodásmódbeli különbség okoz problémát.

A eredménytelenség okai

A BME-VIK elsőéves mérnökinformatikus hallgatók között végzett felmérés azt mutatja, hogy a BME-ProgAlap elvégzése leginkább azoknak okoz problémát, akik a képzésbe belépéskor a feladatokat nem tudják lefordítani adatstruktúrákra, algoritmusokra. Vélhetően e hallgatók absztrakciós készségének hiánya a matematika- és fizikatudásukban is megjelenik.

Az ELTE programtervező matematikus képzés hallgatói közül azoknak okoz nehézséget a tárgy teljesítése, akik korábban nem programoztak. Akik bármilyen nyelven írtak már programot, azoknak nem okozott gondot a struktogram megértése, a specifikáció matematikai nyelvét is tudják olvasni. Akik korábban nem írtak programot, azoknak nem mond semmit az adattípus meghatározása. Az „int” típus a matematikai egész fogalmához kapcsolódik, nem a számítógépes ábrázoláshoz. Például a szövegként megadott egészszám integerré alakításához „beépített függvény”-t keres, nem jut eszébe a szöveg → karakter sorozat → számjegy sorozat → szám átalakítás. Hallgatói visszajelzések alapján, a tanult programozási tételek struktogramját megtanulják lerajzolni és „elhiszik, hogy jelent valamit”. A programozási tételek általános megoldási módot jelentenek, de ehhez nincs gyakorlati tapasztalatuk, nincs miből leszűrni az általános érvényű szabályokat. Programot 2 hónap elteltével csak támogatással tudnak írni, nem hiszik el, hogy működni fog. Az elméleti megállapítások gyakorlati kivitelezése elképzelhetetlen számokra.

Az ELTE-ProgAlap tematikájában jelentős szerepe van a feladatok általános értelmezésének, a modellalkotásnak, a feladat tervezett megoldásának. A sikertelenség legjellemzőbb oka, hogy a hallgatónak nincs kellő gyakorlati tapasztalata, amire az általános megoldást építhetné. 1-2 példából nem következik számára az általános szabály, a tétel.

Összevetve a két ProgAlap célját és eredménytelenségének okait, láthatjuk, hogy az, ami az egyikben hiányzik, az hangsúlyos a másikban. Mindkét képzésre igaz, hogy további félévek során a hiányokat pótolják, azonban ez nem megoldás azon hallgatóknak, akik a hiány miatt nem tudják elvégezni a tárgyat. A lemorzsolódás csökkentése a hiányok előzetes pótlásával lehetséges, azaz mindkét képzéshez programozásból előtanulmányok szükségesek, de más-más területen.

Összefoglalás és konklúzió

Két tantárgyat vizsgáltunk meg:

- az ELTE IK programtervező informatikus képzésének Programozási Alapismeretek tárgyát és
- a BME VIK mérnökinformatikus képzésének A programozás alapjai tárgyát.

A kutatás során megállapítottuk:

1. A két tárgy tartalmilag hasonlónak látszik, de a képzési célban, a fejlesztendő gondolkodási készségben jelentős eltéréseket mutat.

2. Mindkét képzés alapozó, azaz a saját területén nem kíván előzetes ismeretet, ugyanakkor kimutatható, hogy a másik képzésben megfogalmazott alapismereteket elvárja.
3. Mindkét képzésre igaz, hogy a lemorzsolódás egyik jelentős oka a másik képzésben meglevő, az adott képzésből hiányzó alapismeret hiánya.

A fentiekből következik, hogy

- a BME-VIK mérnökinformatikus szakára olyan hallgatókat várnak, akik jó algoritmizálási, absztrakciós készségekkel rendelkeznek és érdeklődésük központjában a „Hogyan működik?" kérdés, a gyakorlati megvalósítás áll.
- az ELTE IK Programtervező matematikus szakára olyan hallgatókat várnak, akiknek van tapasztalatuk, gyakorlatuk kisebb programok írásában és érdeklődésük központjában a nagy programok készítése, programok modellezése áll.

Az egyetemi képzést megelőzően, alapvetően a közoktatásban, de céges illetve magánszervezésben folyó képzésekben is szükség van a programozás széles spektrumú oktatására. A különböző programozás oktatási módszerek eltérő képességeket, készségeket fejlesztenek, de ezek mindegyike szükséges bármelyik képesség illetve készség egyedi fejlesztéséhez.

Praktikusan, egy programozási feladat megoldásának „szépsége", helyessége függ az oktatás módszerének orientációjától, a fejlesztendő képességektől és készségektől. A ProgAlapot megelőző képzésben az oktatási módszer tekintetében nincs királyi út, ellenben minden utat végig kell járni, a programozás oktatásának minden módszerét alkalmazni kell. Eközben be kell mutatni mindegyik szemléletmód alapján a helyes illetve kerülendő megoldásokat. A pályaorientációt befolyásoló szempont, hogy egy-egy diák melyik módszerrel fejleszthető, de a többi módszerrel fejlesztett képességeknek és készségeknek is birtokába kell kerülnie.

Irodalom

[1] European Comission, „CodeWeek.EU," 10 10 2015. [Online]. Available: http://codeweek.eu/. [Hozzáférés dátuma: 20 10 2015].

[2] Code.org, „Hour of Code," 2015. [Online]. Available: https://hourofcode.com/hu. [Hozzáférés dátuma: 20 10 2015].

[3] Asia-Eurpoe Foundation, „<Coding_4_Education> 12th ASEF Classroom Network Conference," 16 11 2015. [Online]. Available: http://www.asef.org/projects/themes/education/3596-12th-asef-classroom-network-conference. [Hozzáférés dátuma: 10 11 2015].

[4] World Robot Olympiad Association Ltd, „World Robot Olympiad," 6 11 2015. [Online]. Available: http://wroboto.org/. [Hozzáférés dátuma: 7 11 2015].

[5] Informatics Europe & ACM Europe Working Group on Informatics Education, „Informatics education: Europe cannot afford to miss the boat," április 2013.. [Online]. Available: http://europe.acm.org/iereport/ACMandIEreport.pdf. [Hozzáférés dátuma: 1. november 2014.].

[6] J. M. Wing, *Computational Thinking and CS@CMU,* Carnegie Mellon University, 2006.

[7] A. Cohen és B. Haberman, „CHAMSA: five languages citizens of an increasingly technological world should acquire," *ACM Inroads,* %1. kötet1, %1. szám4, pp. 54--57, 2010.

[8] Department for Education, GOV.UK, „National curriculum," 16. július 2014.. [Online]. Available: https://www.gov.uk/government/collections/national-curriculum. [Hozzáférés dátuma: 1. novewmber 2014.].

[9] Simon Peyton Jones, Bill Mitchell és Simon Humphreys, „Computing at school in the UK,” 2013. [Online]. Available: http://research.microsoft.com/en-us/um/people/simonpj/papers/cas/computingatschoolcacm.pdf.

[10] M. M. Sysłlo és A. B. Kwiatkowska, „Informatics for all high school students: a computational thinking approach,” *Informatics in Schools. Sustainable Informatics Education for Pupils of all Ages,* %1. kötet7780, pp. 43--56, 2013.

[11] IVSZ, „Az Iskolai digitális oktatás megújítási terve,” 17 06 2015. [Online]. Available: http://ivsz.hu/projektek/digitalis-oktatasi-kialtvany/. [Hozzáférés dátuma: 20 10 2015].

[12] P. Szlávi és L. Zsakó, „Methods of teaching programming 1(2):,” *Teaching mathematics and Computer Science,* %1. kötet01, pp. 247-258, 2003.

[13] Z. Dr. Czirkos, „InfoC – Programozás alapjai I.,” BME EET, 2009-2015. [Online]. Available: https://infoc.eet.bme.hu/. [Hozzáférés dátuma: 12 11 2015].

[14] ELTE IK, „Programozási alapismeretek,” [Online]. Available: http://progalap.elte.hu. [Hozzáférés dátuma: 10 11 2015].

[15] BME VIK, „A programozás alapjai 1 (Tantárgyi adatlap),” 2015. [Online]. Available: https://portal.vik.bme.hu/kepzes/targyak/VIEEAA00/. [Hozzáférés dátuma: 20 10 2015].

[16] „Programozási alapismeretek (Általános információk),” 2015. [Online]. Available: http://progalap.elte.hu/. [Hozzáférés dátuma: 20 10 2015].

[17] Z. Szalayné Tahy és Z. Dr. Czirkos, „Az Informatika más...,” in *DOSZ,* 2015.

[18] Z. Dr. Czirkos, „Programozási tételek,” BME EET, Budapest, 2015.

[19] Z. Dr. Czirkos, „Tömbök algoritmusai. Rekurzió,” BME EET, Budapest, 2015.

[20] G. Dr. Papp, G. Dr. Horváth, P. Szlávi és L. Dr. Zsakó, „Programozási alapismeretek 4. előadás,” ELTE IK, Budapest, 2015.

[21] G. Dr. Papp, G. Dr. Horváth, P. Szlávi és L. Dr. Zsakó, „Programozási alapismeretek 7. előadás (6–7. dia),” ELTE IK, Budapest, 2015.