Programozási alapfogalmak sorrendje a módszeres és funkcionális programozásban

Szabó Zsanett[1], Visnovitz Márton[2]

[1]szabo.zsanett@inf.elte.hu
[2]visnovitz.marton@inf.elte.hu
ELTE IK

Absztrakt. A programozás oktatásában egy adott tanítási módszer, stratégia kiválasztása nagy mértékben meghatározza, hogy a tanulók milyen sorrendben ismerkednek meg a programozás alapfogalmaival. Magyarországon a programozás tanításának egyik leggyakoribb módszere az úgynevezett módszeres (algorithm-first) programozás, de más megközelítések is léteznek, például a funkcionális programozás eszközeivel való tanítás (functional-first). A két módszerben a fogalmak megismerésének sorrendje számos ponton eltér egymástól. Ebben a cikkben a programozás alapfogalmainak megismerési sorrendjét vizsgáljuk és hasonlítjuk össze a módszeres, illetve a funkcionális programozáson alapuló tanítási módszerek esetén.

Kulcsszavak: bevezető programozás, programozás oktatása, programozás alapfogalmai, módszeres programozás, funkcionális programozás

1. Bevezetés

A programozás oktatása az informatikaoktatás egyik legfontosabb része, szinte egyidős a számítástechnikával [1]. Tanítására többféle módszer alakult ki az idők során. A programozás, illetve programozással való problémamegoldás megfelelő és hatékony tanításához, a megfelelő fogalomstruktúra kialakításához elengedhetetlen a témakör és a témakörhöz tartozó ismeretanyagok, fogalmak és módszerek alapos végiggondolása. Ez elengedhetetlen a témakör tudatos felépítéséhez, a tananyag és az órák megtervezéséhez.

A tudatosan felépített tanmenet és tematikus terv elkészítéséhez szükség van arra, hogy a saját fogalomrendszerünket átlássuk, tudatosítsuk. Tanárként tisztában kell lennünk azzal, hogy a témakörhöz tartozó fogalmak közül melyek az igazán fontosak, azok hogyan kapcsolódnak egymáshoz, hiszen célunk az, hogy a diákjaink fogalomrendszerében is kialakuljanak azok a kapcsolatok, összefüggések, amelyek később megkönnyítik majd az újabb és újabb fogalmak beépülését az addigiak rendszerébe.

A különböző tanítási és programozási módszerek több szempontból jelentősen eltérnek egymástól. Az egyik nagy különbséget épp a programozáshoz kapcsolódó alapfogalmak megismerési, illetve elsajátítási sorrendje jelenti, de nem csak ez a különbség. Vannak olyan programozási alapfogalmak, amelyek csak az egyik, vagy csak a másik módszer alkalmazása esetén kerülnek elő vagy kaphatnak nagyobb hangsúlyt. Emiatt érdemes átgondolni, hogy mik ezek a különbségek, hogy milyen előnyök, illetve esetleges hátrányok jelennek meg az egyes tanítási módszerek alkalmazása esetén.

Ebben a cikkben két módszert, a módszeres és a funkcionális programozáson alapuló tanítást elemezzük és hasonlítjuk össze a programozási alapfogalmak szempontjából. A kétféle módszer bemutatása és a programozásoktatásban való használatuk vizsgálata után először külön-külön mutatjuk majd be a módszeres és a funkcionális programozás alapfogalmait tartalmazó fogalomhálót. Vizsgáljuk, hogy az egyes módszerek esetén mely fogalmak kerülnek, illetve nem kerülnek elő. Ezen túl a fogalmak megismerési, elsajátítási sorrendjét is elemezzük a kétféle programozástanítási módszer szerint, majd a két fogalomháló különbségeit, hasonlóságait elemezzük.

A különböző alkalmazási lehetőségek, előnyök és hátrányok miatt a tanítási módszerek általában nem tisztán, hanem vegyesen kerülnek alkalmazásra a tanítási folyamat során. Ennek oka éppen az,

hogy mindkettőnél az előnyöket szeretnénk kihasználni, a hátrányokat pedig próbáljuk elkerülni. A két módszer vizsgálata és összehasonlítása után kitérünk arra is, hogy milyen lehetőségek vannak a két módszer együttes alkalmazására.

2. A módszeres és funkcionális programozás

Programozás tanításáról tulajdonképpen az 1970-es évektől beszélhetünk a mai értelemben. Az áttörést a strukturált programozás néven megjelent módszertan hozta el, melynek lényeges újítása az volt, hogy kevés, de jól definiált algoritmus és adatszerkezet használatára szorítkozott. Az *„oszd meg és uralkodj”* elv alkalmazásával a problémákat részproblémákra osztotta, valamint a programozási nyelvtől elválasztva absztrakt algoritmusokat és adattípusokat vezetett be, ezek alkalmazásával oldotta meg a programozási feladatokat [1,2,3]. Ezt megelőzően a programozás lényege a gépek belső működésének megértése volt, és szinte minden feladat megoldásához új algoritmust kellett kitalálni.

A módszeres programozás, vagyis algoritmusorientált programozási és programozástanítási módszer [1] leírása alapján az algoritmus megtervezését, megértését teszi a programozási folyamat középpontjába. A teljes programozási folyamatot áttekinti a feladat meghatározásától, specifikációjától egészen a dokumentáció elkészítéséig, de a legnagyobb hangsúlyt az algoritmus és az adatstruktúra megtervezése, az algoritmus helyességének belátása kapja. Mind emellett a programozási folyamat további részeiben is megjelennek algoritmusorientált elemek.

Az algoritmusorientált elképzelés egyik alapgondolata, hogy a tervező a végrehajtó szerepébe képzelheti magát, s így az algoritmus helyességéről lehetnek informális elképzelései. Ez egy szekvenciális végrehajtási elképzelésnél nagyon jól építhet egyéni tapasztalatokra, objektumelvű vagy párhuzamos modellekben pedig csoportok működésében, viselkedésében szerzett tapasztalatokra.

Az algoritmus előállításának alapgondolata a szisztematikus felépítés. A Pólya-féle problémamegoldási módszerhez [4] hasonlóan a módszeres programozás a feladatokat általános feladattípusokra, azok megoldásséméira vezeti vissza, az úgynevezett programozási tételekre [5]. A módszeres programozási módszer az *imperatív* programozási elvet követi, melynek két fő jellemzője a parancs és az állapot. Meghatározott parancsokkal változtatja meg a program állapotát, változókkal dolgozik. Az imperatív elv szerint a program készítése során megadjuk a végrehajtandó **lépések egy sorát**, amelyeket el kell végezni ahhoz, hogy a megoldáshoz, a futtatás eredményéhez jussunk.

Mivel a módszeres programozás fókuszában az algoritmus van, így a tanulók elsősorban nem egy konkrét programozási nyelvet, hanem magát az algoritmikus gondolkodásmódot sajátítják el, ismereteik nem lesznek programozási nyelvhez kötöttek [2,6,7,8].

A funkcionális programozás egy, a módszeres programozástól nagyban eltérő paradigma, melynek alapjai matematikai függvényekben, illetve az azokra épülő lambdakalkulusban [9] gyökereznek. Ebben a programozási módszerben a programot egy függvénynek (kifejezésnek) tekintjük, mely függvény kiértékelése tekinthető a program futásának. A program bemenetét a függvény paraméterein keresztül adjuk meg, a program kimenete a kiértékelt függvény által visszaadott érték [10,11].

Típusát tekintve a funkcionális programozás a *deklaratív* programozási módszerek közé tartozik, melyekben a program működésének folyamata helyett a kiszámítás logikájának leírása van a központban [12]. Az ilyen programozásban a programunk nem más, mint definíciók halmaza, mely definíciók alapján a program kimenete megfelelő bemenetek mellett egy kiértékeléssel előállítható. Ennél a módszernél a programozó azt mondja meg a számítógépnek**, hogy mit számítson ki**, ahelyett, hogy megmondaná, **hogyan** tegye.

3. A deklaratív és imperatív megközelítés az informatikaoktatásban

Az előzőkből láthattuk, hogy a módszeres, illetve a funkcionális programozási paradigmák között az egyik legfontosabb különbség, hogy míg a módszeres programozásban *imperatív*, addig a funkcionális programozásban *deklaratív* módon készítjük el a programunkat. Ez a két megközelítés számos más helyen is megjelenik az informatikai képzésben. A deklaratív megközelítéssel találkozhatunk például weboldalak készítésekor, adatbáziskezelésnél vagy táblázatkezelési feladatok megoldásakor. A HTML maga is egy deklaratív leírónyelv, de az SQL lekérdezőnyelvben is egy definíciót adunk meg arra, hogy milyen eredményt szeretnénk látni. Az Excel függvények és képletek alkalmazása tulajdonképpen nem más, mint funkcionális programozás. Az imperatív módszerrel találkozhatunk például prezentációkészítés közben, amikor egy adott dián lévő animációkat állítjuk be, de ide tartozhat a robotok vezérlése, illetve a hétköznapi algoritmusok alkalmazása is.

A kétféle megközelítés nemcsak a feladatmegoldásban jelenhet meg, hanem már egy feladat megadásának módjában is. Ugyanaz a feladat feladható egymás után végrehajtandó utasítások sorozataként, vagyis utasításkövető, imperatív módon; valamint úgy is, hogy csak az elvárt eredményt határozzuk meg megfelelő definíciókkal, leírásokkal, vagyis deklaratív, mintakövető módon. Az ilyen jellegű utasításkövető vagy mintakövető feladatleírások megjelenhetnek számos témakörben, legyen szó dokumentum-, prezentáció- vagy weboldalkészítésről vagy képszerkesztési feladatról (**1. ábra**).

1. Rajzolj egy 2,5 cm × 12,5 cm méretű piros (#FF0000) téglalapot! 2. Rajzolj egy azonos méretű fehér (#FFFFFF) téglalapot, amely a piros téglalaphoz felső élével illeszkedik! 3. Rajzolj egy zöld (#55B24C) téglalapot ugyanolyan méretben, amely a fehér téglalaphoz felső élével illeszkedik!	Készíts magyar zászlót, mely három egymáshoz illeszkedő, azonos méretű (2,5 cm × 12,5 cm) téglalapból áll, a felső téglalap piros, a középső fehér, az alsó zöld színű.

1. ábra: Egy feladat megadása imperatív és deklaratív módon

A tanulóknak mind a kétféle feladatmegadási vagy feladatmegoldási módszert érdemes megismerni, mivel mind az imperatív, mind a deklaratív módszerek megjelennek a hétköznapi életünkben is: legyen szó egy használati, vagy összeszerelési útmutatóról (imperatív) vagy egy felújítási terv elkészítéséről egy lakás felújításához (deklaratív). Mivel ezek a koncepciók a hétköznapi életünkben is erősen jelen vannak, a természetes gondolkodásunk részét képezik, ezért fontos mind a kettő feladatmegoldási módszert megismertetni a tanulóinkkal. Ennek remek eszköze lehet a módszeres és a funkcionális programozás tanítása az iskolában.

4. A módszeres és funkcionális programozás megjelenése informatikai tantervekben

A módszeres és a funkcionális programozás is számos tantervben, tanmenetben, módszerben megjelenik, mint a programozásoktatás eszköze. A különböző képzési szinteken ezeknek a paradigmáknak más-más szerepe lehet az oktatásban, ezért különböző képzési szinteken eltérő formákban jelenhet meg a használatuk.

A felsőfokú programozásoktatásban a módszeres programozás tanítása a mai napig meghatározó. A strukturált és algoritmikus programozással kapcsolatos ismeretek gyakorlatilag minden felsőfokú

programozásképzésben megjelennek, sőt, más, nem informatikai képzések programjában is előfordulnak. Gyakori módszer a programozás bevezetéséhez a módszeres programozás, azon belül is a programozási tételek használata.

A funkcionális programozás mint módszer használata az informatikai iparban erősen meghatározó, ezért magától értetődő, hogy a legtöbb felsőfokú informatikusképzés foglalkozik a témával. Habár a funkcionális programozás a legtöbb egyetemi programozásképzésben megjelenik, intézményenként komoly eltérések vannak abban, hogy a képzési tervben hol jelenik meg ez a paradigma. A programozás alapjai megtaníthatók mind módszeres, mind funkcionális programozás segítségével [7,13], sőt az egyes paradigmák közötti átjárhatóság is biztosított [14]. Éppen ezért a döntés, hogy a programozást melyik paradigma segítségével vezetjük be, módszertani megfontolásokon kell alapuljon, melynek összhangban kell állnia az adott képzés céljaival [15,16].

Bizonyos egyetemek (pl. ELTE, University of Cambridge, University of Oxford) már az első félévtől kezdve tanítják a funkcionális paradigmát a módszeres programozás mellett, így a hallgatók (részben) ezen keresztül (is) ismerkednek meg a programozás alapjaival. Arra is találunk példát a világban, hogy a programozási bevezető kizárólagosan funkcionális nyelvek segítségével történik (pl. Imperial College London), a módszeres programozás alapjaival pedig csak ezt követően ismerkednek meg a hallgatók. Ezzel szemben bizonyos egyetemeken (pl. BME) a funkcionális programozás csak későbbi félévekben jelenik meg, a programozás bevezetésére kizárólagosan a módszeres programozást használják.

Természetesen a kétféle paradigma nem csak a felsőoktatásban kerül elő. A 2020-ban bevezetett Nemzeti Alaptantervben (továbbiakban NAT) [17] leginkább a módszeres programozás alapjai jelennek meg. Olyan kulcsszavakat találunk a dokumentumban, mint algoritmizálás, illetve ahhoz kapcsolódó fogalmak, például *„vezérlési szerkezet”*. A funkcionális programozás alapjait lényegesen nehezebb megtalálni a NAT-ban. A NAT programozással kapcsolatos témaköreiben nem kerül közvetlenül említésre a funkcionális programozás, de a függvények használata megjelenik a matematika tantárgynál, illetve a digitális kultúra tantárgyon belül a táblázatkezelés témakörben is. Ezek alapján mondhatnánk, hogy a közoktatásban a funkcionális programozás mint olyan nem jelenik meg, de a függvények használata a táblázatkezelés témakörben alapvetően nem sokban különbözik a klasszikus funkcionális programozástól. Ezt mi sem mutatja jobban, mint hogy léteznek módszerek, melyek kifejezetten a táblázatkezelésre alapozva, funkcionális programozáson keresztül vezetik be a tanulókat a programozás világába [18], majd ezt követően az így szerzett tudást használják fel az algoritmikus gondolkodás fejlesztésére [19].

Az érettségi vizsgán az aktuális, 2012-es NAT-on [20] alapuló követelményrendszer [21] szerint programozási feladat csak emelt szinten van, de egyszerű függvények használata a táblázatkezelés és adatbáziskezelés feladatnál is elvárás. Habár emelt szinten alapvetően a módszeres programozás szerint vannak meghatározva a követelmények (pl. ismerje a mondatszerű algoritmus-leíró eszközt; tudja használni az elemi programozási tételeket; legyen képes egy mondatszerű leírással készült algoritmust a használt programozási nyelvben kódolni), a programozási feladat egy-két dolgot leszámítva (például fájlból való beolvasás és kiírás) tisztán funkcionális programozással is megoldható.

A 2020-as NAT-on alapuló, 2024. január 1-tól érvényes érettségi követelmények [22] alapján már középszinten is elvárás az algoritmizálás és a programozás, de az eljárások és függvények ismerete csak emelt szinten elvárás, középszinten nem. (Korábban emelt szinten sem szerepelt a követelmények között.) A követelmények a módszeres és a funkcionális programozás használhatóságát illetően nem változtak.

Az érettségi feladatsorok programozás feladatai fájlból olvasással kezdődnek, majd a beolvasott adatok feldolgozása után az adatokra vonatkozó kérdések következnek, melyek általában egy-egy programozási tétellel vagy azok összeépítésével oldhatók meg. A 2020-as NAT-on alapuló, 2024-től bevezetésre kerülő vizsgakövetelmények szerint készített mintafeladat alapján az új feladatsorokban a

korábbiaknál összetettebb kérdésekre lehet majd számítani emelt szinten, de továbbra is igaz, hogy a feladatok megvalósíthatók a programozási tételek alkalmazásával és azok összeépítésével. A programozási tételek módszeres és funkcionális módon is megvalósíthatók [14], így a feladatok megoldása mind a két paradigma használatával lehetséges.

A digitális kultúra tantárgy 2024-től bevezetésre kerülő érettségikövetelmények szerint készített középszintű mintafeladatsorának [23] programozási feladatára viszont már nem mondható el ugyanez. A mintafeladatban a program készítésének lépései imperatívan, lépésről lépésre vannak meghatározva. Konkrét feladatként szerepel bizonyos (előre meghatározott nevű) változók létrehozása, értékadások, valamint ciklus és elágazás létrehozása is. Emellett szerepel véletlen számok generálása is, amihez viszont mindenképp szükség van a véletlen számot generáló függvény alkalmazására. A feladatmegoldás során végrehajtandó konkrét utasítások miatt a középszintű mintafeladat alapvetően nem oldható meg funkcionálisan.

5. Programozási fogalmak sorrendje és egymásra épülése

A számunkra, illetve a tanított csoport számára megfelelő programozástanítási módszer kiválasztásához fontos végiggondolni egyebek mellett azt is, hogy a különböző módszerek választása esetén mely programozási alapfogalmakat sajátíthatják el a tanulók, hogy mi a programozás tanításának célja az adott csoport esetén, hogy mennyi idő áll rendelkezésre a témakör tanításához.

A fogalmak összegyűjtéskor szembesülnünk kell azzal, hogy az algoritmizálás, programozás témakör rengeteg fogalmat használ, a fogalmak közötti kapcsolatok viszont sok esetben nem egyértelműek. Nem, vagy csak nagyon nehezen határozható meg a fogalmak tanításának *„helyes”* sorrendje, és egy ilyen *„helyes”* sorrend pedig számos tényezőtől függ. Az egyik tényező az, hogy milyen környezetben, milyen programozási nyelv tanulása közben kerülnek elő ezek a fogalmak.

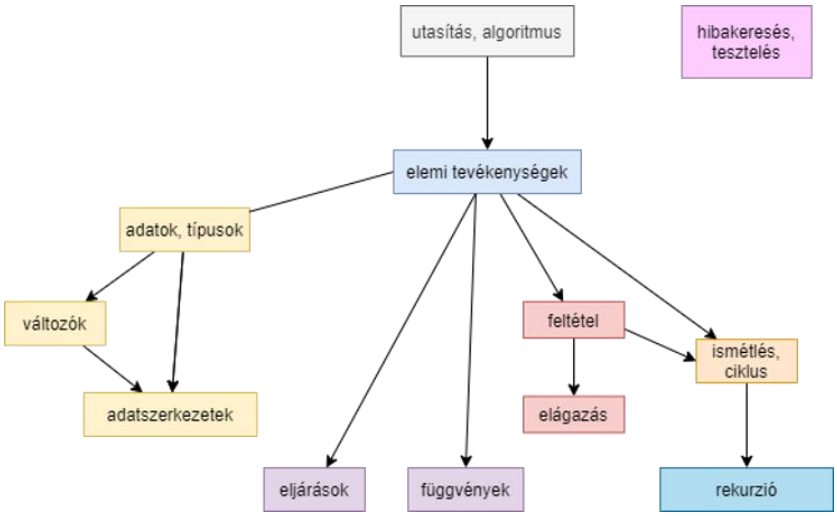
Az algoritmizálás, programozás témakörhöz tartozó alapfogalmak egyfajta általános rendszerezését egy korábbi kutatás [5] mutatja be. A teljes fogalomstruktúra meglehetősen komplex, ezért ebben a cikkben nem a konkrét fogalmak, hanem nagyobb fogalomcsoportok kapcsolatát és tanítási sorrendjét vizsgáljuk majd a módszeres és a funkcionális programozás esetében.

Minden fogalomcsoporthoz több fogalom tartozik, melyek kapcsolatai, tanítási sorrendjük szintén fontos, de a módszeres és funkcionális programozás általunk szemléltetni kívánt különbségei már a fogalomcsoportok szintjén is jól láthatók. Alaposabb vizsgálat esetén további különbségek figyelhetők meg a kétféle módszer fogalomstruktúrájában az egyes fogalomcsoportokon belül.

A fogalomhálókban a fogalomcsoportok közötti kapcsolatok mellett azok tanítási sorrendjét is igyekeztünk szemléltetni. Minél lejjebb került egy fogalom a kapcsolatrendszereket bemutató ábrákon, várhatóan annál később kerül elő a tanulási, tanítási folyamatban. A fogalomcsoportok tanítási sorrendje nem teljesen rögzített. Néhány esetben, a fogalmak közötti kapcsolatokból adódó sorrend kötött, de ezek mellett több fogalomcsoport tanítási sorrendje módosítható a tanított csoport igényeinek és a tanítás céljának megfelelően.

Módszeres programozás

Módszeres programozás esetén a fogalomcsoportokba szervezett fogalomháló a **2. ábra** szerint alakul.



2. ábra: A programozás fogalomcsoportjainak kapcsolatai módszeres programozásban

A módszeres programozás fogalomhálójában, ahogyan ezt a módszer másik neve, az algoritmusorientált programozás is mutatja, az elsőként megismerésre kerülő fogalomcsoport az *utasítás*hoz és magához az *algoritmus*hoz köthető. Az utasítás és algoritmus fogalmaihoz elsőként az *elemi tevékenységek* (pl.: beolvasás, kiírás) kapcsolódnak. A legtöbb nyelvben és fejlesztőkörnyezetben egy új „üres" program automatikusan tartalmaz egy kiírást. Az elemi tevékenységekhez szorosan kötődnek az *adatok, típusok*, azokhoz pedig a különböző *változók* és *adatszerkezetek*.

Az elemi tevékenységekhez kapcsolódó másik fogalom, ami általában nagyon hamar előkerül, az a *feltétel*, hiszen feltétel szükséges az *elágazások* és *ciklusok* használatához is. A fogalomstruktúra egyik legfontosabb és a fogalomcsoporthoz kapcsolódó fogalmak számát tekintve egyik legnagyobb részét képezik a *feltétel*ekhez, *elágazás*hoz és a *ciklus*okhoz kapcsolódó fogalmak. Ezek tisztázása, illetve megértése különösen fontos és elengedhetetlen része a témakör tanulásának, illetve tanításának, hiszen az algoritmusok és programok ezekből a vezérlési szerkezetekből építkeznek.

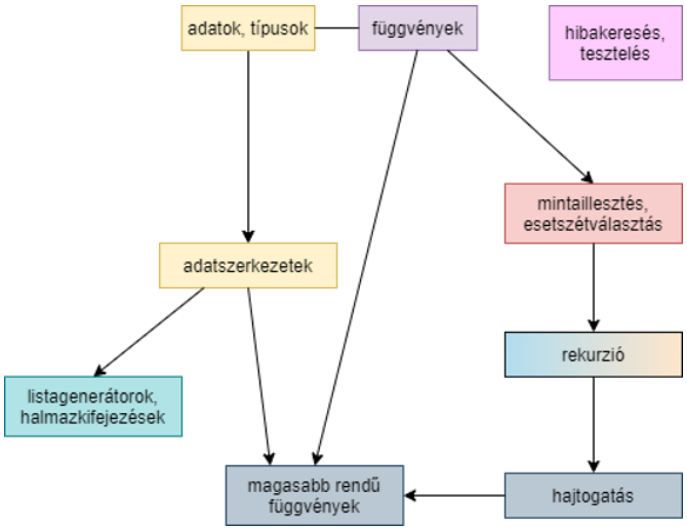
Az *eljárások* és *függvények* a módszeres programozási módszer esetén általában csak később, az alapvető vezérlési szerkezetek használatának begyakorlása után kerülnek elő, ha egyáltalán előkerülnek. Ezek csupán a kódszervezés eszközei, az elkészített program hatékonyabbá, átláthatóbbá tételét segítik, de minden megoldható nélkülük. Elegendő csak a főprogramhoz tartozó függvényt/eljárást használnunk, amiről a programozással még épp csak ismerkedő tanulóknak nem is kell feltétlenül tudniuk, hogy az egy függvény/eljárás. Kezdetben számukra csak annyi a fontos, hogy minden utasítást, amit végre szeretnének hajtani, azt a főprogramon (pl. main függvényen) belülre kell írniuk. Ha kevés a témakör tanítására fordítható idő, akkor az is előfordul, hogy nem is találkoznak olyan feladatokkal a tanulók, ahol függvényeket, eljárásokat használnak, vagy csak egy-egy példán keresztül kerül bemutatásra, mint fejlesztési lehetőség.

A *rekurzió*, a függvényekhez és eljárásokhoz hasonlóan középiskolában érdekességként kerül csak elő a módszeres programozásoktatásban. Ez nem szerves és elengedhetetlen része a témakör tanításának, az alapvető programozási ismeretek elsajátításának.

A programok futtatásakor és az elkészített algoritmusok ellenőrzésekor *hibakeresés*sel és *tesztelés*sel is foglalkoznunk kell. Ezek végigkísérik a teljes programozástanulási folyamatot, de közvetlenül nem kapcsolódnak a többi programozási alapfogalomhoz. Az ide tartozó fogalmak és hibakeresési módszerek nagy mértékben függnek a programozási környezettől, illetve a témakör tanításának mélységétől, de minimális szintű hibakezelésre a választott környezettől függetlenül szükség van már a legegyszerűbb programok elkészítése során is (pl. kódolási hibák javítása).

Funkcionális programozás

A funkcionális programozás legfontosabb fogalma a *függvény,* a paradigma a nevét is innen kapja (angolul: function). A funkcionális programozásban majdnem minden további koncepció közvetve vagy közvetlenül ehhez a központi fogalomhoz köthető, így értelemszerűen ez áll a fogalmak rendszerének központjában (**Hiba! A hivatkozási forrás nem található.**), illetve a sorrendiséget tekintve nagyon korán szerepel. A programozási függvényfogalom bevezetéséhez felhasználhatók korábban megszerzett előismeretek, mint például a matematikai függvényfogalom vagy a táblázatkezelésben használt függvények ismerete.



3. ábra: A programozás fogalomcsoportjainak kapcsolatai funkcionális programozásban

A másik fontos alapfogalom a funkcionális programozásban a típus, mely szintén szorosan köthető a függvényfogalomhoz. A klasszikus, tisztán funkcionális nyelvekben minden típus valójában a függvénynek egy altípusa, de nem feltétlenül szükséges a típusfogalmat ilyen megközelítésben bevezetni. Az egyszerű típusok (logikai, szám, karakter, stb.) önmagukban is bevezethetők és később köthetők a függvényekhez a bemeneti paraméterek típusa és a visszatérési típus révén. Már ezen a ponton komoly különbségek lehetnek a fogalmak bevezetésében a programozási nyelv választásától függően. Egy tisztán funkcionális, erősen típusos nyelv (pl. Haskell) esetén nem tudunk saját függvényt definiálni anélkül, hogy típusokat adnánk meg a függvény szignatúrájában, míg egy dinamikusan vagy gyengén típusos nyelv (pl. JavaScript) esetén lehetőségünk van erre.

A további fogalmi építkezést szintén erősen befolyásolja a nyelvválasztás. Számos nyelv rendelkezik olyan szintaktikai megoldásokkal, mellyel könnyen bevezethető a *mintaillesztés* és az *esetszétválasztás* fogalma, melyek a módszeres programozásban az elágazások analógiái. Ezen fogalmak ismeretében lehetőségünk nyílik megismerni a rekurzió fogalmát, melynek – azon túl, hogy egy általánosságban is

fontos feladatmegoldási módszer – a funkcionális programozásban, a komplex adatszerkezetek feldolgozásában is jelentős szerepe van. Komplex adatszerkezetek (pl. lista) rekurzióra épülő feldolgozását jellemzően a *hajtogatás* módszerének segítségével tudunk megtenni.

A mintaillesztés és a rekurzió mellett egy másik vonalon, a *típus* és az ehhez szorosan kapcsoló *adat* fogalmainak bevezetése után lehetőség van továbblépni a haladóbb, *összetettebb adatszerkezetekre*, mint például a rendezett n-es, a lista vagy éppen a rekord. Ezek az összetett típusok már lehetőséget adnak az adatmodellezéssel kapcsolatos ismeretek elsajátítására, elmélyítésére, és utat nyitnak a funkcionális programozás bizonyos haladóbb lehetőségei felé. Ezen haladó lehetőségek közé tartoznak például bizonyos nyelvekben (pl. Clean, Haskell) a *listagenerátorok*, illetve *halmazkifejezések*, melyekkel deklaratív módon tudunk összetett adatszerkezet-példányokat definiálni.

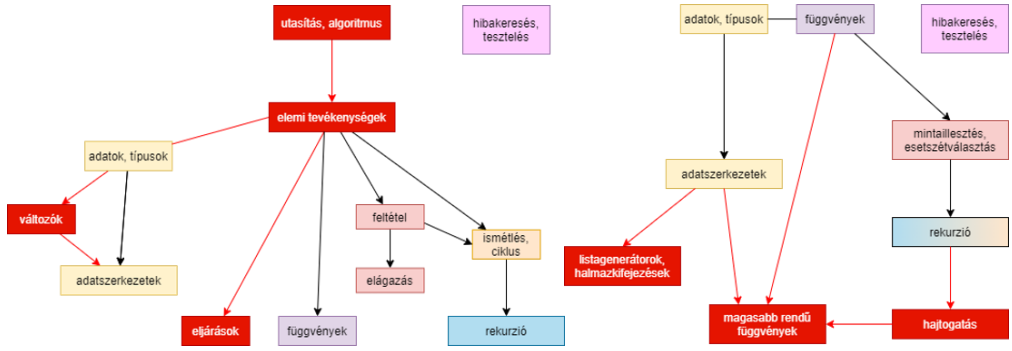
Egy másik irány, amerre tovább haladhatunk az összetett adatszerkezetektől, az a magasabb-rendű függvények világa. Ezen a ponton már szükségessé válik a függvény mint elsőrendű típus fogalmának bevezetése, ha ez korábban nem történt meg. A magasabb rendű függvények igencsak hatékony eszközök a feladatmegoldásban. Segítségükkel lehetőségünk nyílik komplex adatfeldolgozási feladatok megoldására oly módon, hogy a feldolgozás logikája helyett (pl. hogyan iterálunk végig egy listán) a konkrét megoldandó feladat specifikus részeire tudunk koncentrálni (pl. egy kiválogatás esetén mi a kiválogatás feltétele).

A magasabb-rendű függvényekig a hajtogatáson, és a magasabb-rendű függvények implementálásán keresztül vezető úton is eljuthatunk. Szinte minden, a funkcionális programozást támogató nyelvben előre definiálva megjelennek ezek a magasabb-rendű függvények, melyek közül a leggyakoribbak (reduce/fold, map, filter, find, any/some, all/every stb.) megfelelnek az algoritmikus programozásban tanított programozási tételeknek [14].

Természetesen a módszeres programozáshoz hasonlóan itt is végigkísérik a fogalmi hálót a teszteléssel és hibakereséssel kapcsolatos ismeretek. Habár ezek módja sok helyen eltér a módszeres programozásban alkalmazott módszerektől, a kapcsolódó fogalmak (pl. szintaktikus, szemantikus hiba) hasonlóak.

6. Fogalmak a módszeres és funkcionális programozásban

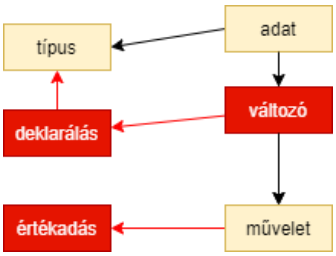
A módszeres és a funkcionális programozáshoz tartozó fogalomhálók ismertetése alapján is könnyen belátható, hogy jelentős különbségek vannak a kétféle módszer alkalmazása esetén előkerülő fogalmak között, illetve a fogalmak tanítási sorrendjében. A 4. ábra a két fogalomháló összehasonlítását segíti. Egymás mellett látható a két fogalomháló és pirossal jelöltük azokat a fogalomcsoportokat, amelyek az egyikben vagy a másikban nem jelennek meg.



4. ábra: A módszeres és a funkcionális programozási módszerek fogalomhálóinak összehasonlítása fogalomcsoportok szintjén

A funkcionális programozás fogalomhálójában nem kerül elő az utasítás és algoritmus, az elemi tevékenységek, a változók és az eljárások, a módszeres programozás fogalomhálójából pedig a magasabb rendű függvények, listagenerátorok és halmazkifejezések, valamint a hajtogatás hiányzik a funkcionális programozás fogalomhálójához képest.

A két fogalomháló között nemcsak fogalomcsoportok szintjén, hanem az egyes fogalomcsoportokon belül is vannak különbségek. Ebből következően a fogalomhálók azon részei sem teljesen egyeznek meg, amelyeket a fogalomcsoportok struktúrája alapján azonosnak gondolunk. Az 5. ábra segítségével példaként megmutatjuk, hogy az *adatok, típusok* fogalomcsoport hogy néz ki fogalmak szintjén. Módszeres programozás esetén mind a hat fogalom (*adat, típus, változó, deklarálás, művelet, értékadás*) megjelenik, de funkcionális programozás esetén ezek közül csak három, az *adat*, *típus* és *művelet* kerül elő, mert a klasszikus, tisztán funkcionális programozásban nincsenek változók, így azokat deklarálni sem lehetséges, illetve értéket sem tudunk adni nekik.



5. ábra: Az *adatok, típusok* fogalomcsoportban megjelenő különbségek

7. Lehetőségek a paradigmák kombinálására

Ahogyan az előzőkből láthattuk, a sorrendiségen túl a megismert tartalmak terén is komoly különbségek vannak attól függően, hogy a programozás bevezetésére módszeres vagy funkcionális programozást választunk. Ezek alapján jogosan merül fel a kérdés, hogy mit tehetünk, ha mind a két paradigma fogalmaival szeretnénk megismertetni a tanulókat.

Kézenfekvő megoldás lehet a paradigmák egymás utáni bemutatása. Számos helyen találunk példát erre a megközelítésre, legyen szó egyetemi képzésről vagy iskolai módszertanról. Ha ezt a módszert választjuk, lehetőségünk van kihasználni a fogalmi hálók közötti átfedéseket, ezeket kapcsolódási vagy akár kiindulási pontként használva mikor bevezetjük a diákokat a második paradigma világába. Ennek a megközelítésnek egy lehetséges veszélye, hogy az elsőnek tanult paradigma módszerei, gondolkodásmódja annyira meghatározóvá válik a tanuló problémamegoldásában, hogy a másodikként tanult

paradigma használata közben is ezeket a betanult módszereket akarja majd használni, emiatt pedig nehezebbé válik számára a második paradigma elsajátítása.

Egy másik lehetséges megközelítés – amennyiben lehetőség van rá – a módszeres és a funkcionális programozás párhuzamos tanítása (pl. azonos félévben levő egyetemi kurzusok esetén). Ez a megoldás segíthet abban, hogy egyik paradigmát se tekintse a tanuló „elsődlegesnek" csupán azért, mert azzal ismerkedett meg először. Ha ezt a megoldást választjuk, akkor ügyelni kell arra, hogy a kétféle programozási paradigma tanítása megfelelően elkülönüljön egymástól, ne keveredjenek a különálló részek, de fontos a kapcsolódási pontok bemutatása is. Ezzel a módszerrel, amennyiben a két irányzat tanítása formálisan is elkülönül (pl. külön kurzus formájában), jellemzően két különböző programozási nyelven keresztül történik a paradigmák bemutatása. Mind az algoritmikus, mind a funkcionális programozáshoz van lehetőségünk olyan nyelvet választani, mely kiemeli az adott paradigma sajátosságait, amin keresztül annak jellegzetességei könnyen bemutathatók.

A módszeres és a funkcionális programozás kombinálására egy harmadik lehetőség az, hogy olyan programozási nyelvet választunk, ami támogatja a feladatmegoldást mindkét módszerrel (pl. Python, JavaScript). Egy ilyen multiparadigmás nyelv használatával nem csak arra nyílik lehetőségünk, hogy egy eszközkészlettel mutassuk be külön-külön a két paradigmát, hanem arra is, hogy a kettőt kombináljuk. Ezáltal lehetőségünk nyílik annak hangsúlyozására, hogy az, hogy többféle programozási paradigma létezik, az nem jelenti azt, hogy egy feladat megoldásában kizárólagosan az egyik vagy a másik mellett kell dönteni. Az ilyesfajta kombinálásra kiváló lehetőség lehet például a magasabb rendű függvények használata, melyeknél a paraméterként átadott függvényt implementálhatjuk akár algoritmikus módon is, vagyis magas szinten funkcionális programozást használhatunk, míg alacsony szinten módszereset. Egy ilyen megközelítés esetén a kétféle paradigmához tartozó fogalmi hálók összefonódnak, a kapcsolódási pontok mentén egy nagyobb, komplexebb fogalmi rendszert kapunk, melyben megtalálhatók a módszeres és a funkcionális programozás fogalmai is. Természetesen itt is nagyon fontos a programozási nyelv kiválasztása, mivel az meghatározza, hogy milyen lehetőségek állnak rendelkezésünkre a feladatmegoldásban. Vannak olyan, kifejezetten a funkcionális programozáshoz köthető megoldások (pl. mintaillesztés, listagenerátorok) melyek a multiparadigmás nyelvekben ritkán fordulnak elő. Értelemszerűen, ha a választott nyelv ezeket nem támogatja, akkor ezek bemutatására, tanítására sincs lehetőség ugyanabban a környezetben, amelyben a többi alapfogalmat bemutatjuk.

8. Konklúzió

Összegzésként elmondható, hogy a módszeres és funkcionális programozási módszer egyaránt fontos szerepet játszik a programozás tanításában. Mindkét módszerre láthatunk példát az oktatás különböző szintjein. Van, ahol csak az egyik vagy csak a másik kerül elő, de az egyetemi képzés során a tanulók jellemzően mindkét módszerrel megismerkednek.

A kétféle módszer nem csak a programozásoktatásban, hanem a hétköznapi életünkben is jelen van, hiszen gyakran találkozhatunk a módszeres programozásra jellemző imperatív feladatmegadással, ahol lépésről lépésre meg van határozva, hogy mit kell tennünk (pl.: automaták használata, szerelési útmutatók). Emellett pedig az is gyakori, hogy a funkcionális programozásra jellemző deklaratív feladatmegadással találkozunk, ahol csak az elvárt eredmény van meghatározva (pl. legyen tiszta a lakás – mit értünk tiszta lakás alatt).

A programozás tanításához választott módszer (módszeres vagy funkcionális) meghatározza, hogy mely fogalmakkal ismerkednek meg a tanulók és a fogalmak elsajátításának sorrendjét is befolyásolja.

Jelentős különbségek vannak a két módszer alkalmazása esetén felépülő fogalomhálóban már fogalomcsoportok szintjén is, de az egyes fogalomcsoportokon belül, fogalmak szintjén is vannak további eltérések.

Az egyetemi képzések felépítésekor fontos módszertani kérdés, hogy milyen sorrendben jelenjen meg a módszeres és a funkcionális paradigma, többféle megoldásra mutattunk példát. Középiskolában az érettségi által meghatározott kimeneti követelmények miatt alapvetően a módszeres programozásoktatás jellemző, de van ellenpélda is, illetve olyan is, hogy már középiskolában találkoznak mindkét módszerrel a tanulók. Középiskolák esetén a tanárok, iskolai munkaközösség felelőssége, hogy a tanított csoportnak megfelelő módszert válasszák.

A középiskolai tanárok, illetve azok számára is nagy segítséget jelenthet, ha kombinálni tudják a módszeres és funkcionális programozás módszereit, akik mindkettő fogalmaival és előnyeivel meg szeretnék ismertetni a tanítványaikat. Erre jó lehetőséget biztosítanak a multiparadigmás környezetek. Ilyen környezet választása esetén a két módszer fogalomhálójának összekapcsolásával szélesebb körű ismeretek adhatók át, de megmarad annak lehetősége is, hogy a tanított csoportnak megfelelő mélységben és részletességgel tárgyaljuk a programozás témakör bizonyos részeit, fogalmait.

Irodalom

[1] Szlávi, P., Zsakó, L.: *Az informatika oktatása*. (2011)

[2] Dijkstra, E. W.: *A discipline of programming*. Prentice-Hall, (1976)

[3] Dahl, O.-J., Dijkstra, E. W., Hoare, C. A. R.: *Structured Programming*. Academic Press: New York, (1972)

[4] Pólya, G.: *A gondolkodás iskolája*. Akkord Kiadó, (2000)

[5] Szabó, Z.: *Problem Solving and Interrelation of Concepts in Teaching Algorithmic Thinking and Programming*, In: Proceedings of the 11th International Conference on Applied Informatics (ICAI 2020), pp. 318–327, (2020), [Online] http://ceur-ws.org/Vol-2650/paper33.pdf

[6] Szlávi, P., Zsakó, L.: *Módszeres programozás: Programozási bevezető*. In: Mikrológia vol. 18. ELTE TTK Általános Számítástudományi Tanszék, Budapest, (1994)

[7] Szlávi, P., Zsakó, L.: *Módszeres programozás: Programozási tételek*. In: Mikrológia vol. 19. ELTE Informatikai Kar, (2008)

[8] Fóthi, Á.: *Bevezetés a programozáshoz*. ELTE Eötvös Kiadó: Budapest, (2005)

[9] Church, A.: *An unsolvable problem of elementary number theory*, In: American Journal of Mathematics, vol. 58, pp. 345–363, (1936), DOI: 10.2307/2268571

[10] Horváth, Z.: *A funkcionális programozás nyelvi elemei*. In: Programozási nyelvek (szerk. Nyékiné Gaizler, J.). Kiskapu kiadó, (2003), p. 56

[11] Barendregt, H., Barendsen, E.: *Introduction to lambda calculus*, In: Nieuw archief voor wisenkunde, (2000)

[12] Lloyd, J. W.: *Practical Advantages of Declarative Programming*, (1994).

[13] Abelson, H., Sussman, J., Sussman, J.: *Structure and Interpretation of Computer Programs*. MIT Press, (1984)

[14] Visnovitz, M.: *Classical Programming Topics with Functional Programming*, In: Central-European Journal of New Technologies in Research, Education and Practice, (2020), DOI: 10.36427/cejntrep.2.2.965

[15] Chakravarty, M. M. T., Keller, G.: *Educational pearl: The risks and benefits of teaching purely functional*

programming in first year, In: Journal of Functional Programming, (2004), DOI: 10.1017/S0956796803004805

[16] Joosten, S., van den Berg, K., van der Hoeven, G.: *Teaching functional programming to first-year students*, In: Journal of Functional Programming, vol. 3, no. 1, pp. 49–65, (1993), DOI: 10.1017/S0956796800000599

[17] *Nemzeti Alaptanterv 2020*. (2020)

[18] Csernoch, M., Bíró, P.: *Sprego Programming*, In: Spreadsheets in Education, vol. 8, no. 1, (2015)

[19] Csernoch, M., Bíró, P., Máth, J.: *Developing Computational Thinking Skills with Algorithm-Driven Spreadsheeting*, In: IEEE Access, vol. 9, (2021), DOI: 10.1109/ACCESS.2021.3126757

[20] *Nemzeti Alaptanterv 2012*. (2012)

[21] Oktatási Hivatal: *Informatika részletes érettségi vizsgakövetelmények (2017. május-júniusi vizsgaidőszaktól)* https://www.oktatas.hu/pub_bin/dload/kozoktatas/erettsegi/vizsgakovetelmenyek2017/informatika_vk.pdf

[22] Oktatási Hivatal: *Digitális kultúra részletes érettségi vizsgakövetelmények (2022. január 1-től)*. (2021)

[23] Oktatási Hivatal: *Digitális kultúra középszintű írásbeli érettségi mintafeladatok a 2024. január 1-től bevezetésre kerülő vizsgakövetelmények szerint*.