PROGRAMOZÁS TANÍTÁSI MÓDSZEREK

METHODS IN TEACHING PROGRAMMING

Szlávi Péter, Szlavi@ludens.elte.hu
Zsakó László, Zsako@ludens.elte.hu
ELTE TTK Informatika Szakmódszertani Csoport

Abstract

Programming methodology is one of the oldest fields of IS education, and thus various methods have evolved for its teaching. While some of them could be used effectively in primary or secondary education, others are more suited to students in higher education. The methods themselves determine the structure and curricula of courses such as *Programming methodology*, *Data types and algorithms*, *Programming technology*.

Összefoglaló

A programozási módszertan az informatika oktatása egyik legrégibb területe, így többféle, ma is használatos módszer alakult ki tanítására. Ezek közül egyesek az általános, illetve középiskolai oktatásban használhatóak eredményesen, mások pedig a felsőoktatásban. Mindegyik módszer alapgondolatából következik, hogy milyen legyen a *Programozási módszertan*, az *Adatszerkezetek és algoritmusok*, a *Programozási technológia* stb. tárgyak tanterve, felépítése.

PROGRAMOZÁS TANÍTÁSI MÓDSZEREK

METHODS IN TEACHING PROGRAMMING

Szlávi Péter, Szlavi@ludens.elte.hu
Zsakó László, Zsako@ludens.elte.hu
ELTE TTK Informatika Szakmódszertani Csoport

Előadásunkban a legelterjedtebb módszereket tekintjük át:

- módszeres, algoritmusorientált;
- adatorientált;
- specifikációorientált;
- feladattípus-orientált;
- nyelvorientált;
- utasításorientált;
- matematikaorientált;
- hardverorientált.

Megjegyezzük azonban, hogy a legtöbb oktató nem tisztán egy módszer mellett kötelezi el magát, hanem vegyesen használ több módszert.

1. Módszeres, algoritmusorientált

Ez a módszer, mint sok másik, a programkészítés teljes folyamatát tekinti át:

- feladat-meghatározás, specifikáció;
- algoritmus- és adatstruktúra tervezés, az algoritmus helyességének belátása;
- kódolás;
- tesztelés;
- hibakeresés, hibajavítás;
- hatékonyság-vizsgálat, minőség-vizsgálat;
- dokumentálás.

Az egyes résztevékenységekkel önállóan, külön-külön kell foglalkozni. Mindegyikben át kell tekinteni az ahhoz a témához kapcsolódó eszközöket és módszereket. Az egyes résztevékenységek közül ebben a módszerben az *algoritmus előállítás*át tekintjük elsődlegesnek, ez szerepel a tanítás során a legnagyobb súllyal, sőt a többi részben is megjelennek algoritmus-orientált elemek.

Az algoritmus előállítás alapgondolata a szisztematikus felépítés. Első lépése az általános feladattípusok és azok általános megoldássémái, az ún. *programozási tételek*. (Formálisan is bizonyítható [SzZs1], hogy az egyes általános feladattípusok általános megoldássémái helyes megoldásai a nekik megfelelő feladatnak, a módszer azonban a formális bizonyítást informálissal helyettesíti.)

Második lépésként konkrét feladatok *programozási tételekre* való *visszavezetés*ével kell foglalkozni, azaz megállapítani, hogy hogyan alkalmazhatók az egyes programozási tételek. Ebben a módszer specialitása, hogy párhuzamosan vizsgálja a specifikációt és az algoritmust, mindkettőben megnézve, hogy hol és mit kell aktualizálni.

Harmadik lépésként foglalkozhatunk programozási tételek összeépítésével, azaz olyan feladatokkal, ahol több programozási tételt kell egyszerre (nem lineárisan egymás után) alkalmazni. Itt is fontos, hogy nemcsak a specifikációra, hanem az algoritmusok összeépítésére is megadjuk a megfelelő (programtranszformációs) szabályokat. [HSzZs]

Az adatstruktúra tervezés (a hagyományos *Adatszerkezetek és algoritmusok* tantárgy egyik anyaga) itt a programozási módszertan szempontjai szerint kerül feldolgozásra, azaz az adatszerkezet egy típus, specifikálással, a struktúra reprezentálásával és a műveletek implementálásával. [PSzZs]

A kódolási fázisban nagyon sok, konkrét vagy általánosabb programozási nyelvtől függő döntésre kerülhet sor. Ez a módszer e döntések egy jelentős részét szintén algoritmikus szemlélettel fogja meg. Így tárgyalja például az iteratív és a rekurzív algoritmusok közöttiátalakítás módszereit [SzZs2], illetve a szokásos programozási struktúraféleségek (elágazások, ciklusok, eljárások, függvények, operátorok, …) egymással történő megvalósítását. A módszer ezen kívül ebben a fázisban foglalkozik a felhasználóbarát programozás eszközeivel, módszereivel (pl. menük megvalósítása). A tesztelési módszereken belül egyaránt foglalkozik a specifikációra épített (ún. fekete doboz) és az algoritmusra épített (ún. fehér doboz) módszerekkel. [SzZsT]

A módszer hatékonyságvizsgálatot is az algoritmusra építve tárgyalja. Ez azt jelenti, hogy a végrehajtási idő, a helyfoglalás és a bonyolultság szempontjából általános feladatosztályokat fogalmaz meg (hasonlóan a programozási tételekhez), s megadja a hatékonyabbra írás algoritmikus sémáit, ötleteit (pl. a sorozat részekre osztása elve alkalmazható a logaritmikus keresés, a quicksort rendezés, a párhuzamos maximum- és minimum-kiválasztás algoritmusainál éppúgy, mint mondjuk az intervallumfelezéses gyökkeresésnél). [Zs]

Mivel itt (és a következő két módszerben) a programozási nyelv nem játszik elsődleges szerepet, így a programozási ismeretek felépítését nem befolyásolja túlzottan, az így tanult programozók nem lesznek nyelvhez kötöttek.

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.

2. Adatorientált

Ez a módszer nagyon hasonlít az előzőre, az algoritmus-alkotás helyett azonban az *adatstruktúrá*t, a *típusfinomítás*t tekinti elsődlegesnek. Alapgondolata, hogy a feladat-meghatározást típus-specifikációként tekinti és a típusfinomításhoz rendel algoritmikus struktúrákat:

- direkt szorzat – szekvencia;
- unió, alternatív adatstruktúrák – elágazás;
- sokaság (halmaz, sorozat, hierarchikus és hálós szerkezetek)– ciklus;
- rekurzívan definiált sokaság (adatrekurzió, vagy rekurzív típus)– rekurzió.

A módszer tiszta változatában a felhasználóval való kapcsolat is típusfinomításként fogható fel. Ebből kapjuk a bemeneti (űrlapok) és kimeneti (jelentések) formátumokat. Az előző módszer *programozási tétel* fogalmával szemben itt az *adatfeldolgozási típusfeladatok* kerülnek előtérbe (adatfelvitel, listázás, összegfokozatos listázás, másolás, időszerűsítés, …). Ezekre ugyanúgy elkészíthetők az általános adatstruktúrák és algoritmusok, mint a programozási tételekre. [SzZs3]

A módszer jellegénél fogva más feladattípusokat helyez előtérbe, mint az előző. Az algoritmusorientált módszerre jellemző, hogy először az algoritmikus struktúrák kiterjesztésével foglalkozik, azaz alapelve az *egyszerű adatstruktúra – összetett algoritmikus struktúra* elv. Az adatorientált elképzelés ezzel szemben az *összetett adatstruktúra – egyszerű algoritmikus struktúra* elvet követi. Itt a programok nagyon sokáig például az „1 olvasás – feldolgozás – 1 írás” alapelvre épülhetnek. [B]

3. Specifikációorientált

A módszer alapgondolatában az első kettőhöz hasonló. A programozási folyamat leghangsúlyosabb részének azonban a *formális specifikálás*t tartja, a specifikációból automatikusan vezeti le az algoritmust, majd merev kódolási szabályok segítségével állítja elő a kódot. Az elsőhöz hasonlóan itt is beszélhetünk programozási tételekről, ezek algoritmusai azonban a megadott specifikációból való levezetéssel születnek. [F]

Míg az algoritmusorientált módszerben az algoritmus átalakítása játssza a főszerepet, addig itt sokkal hangsúlyosabb a specifikáció átalakítása. Ennek megfelelően az adatszerkezetek és algoritmusok témakör is sokkal elméletibb ismereteket tartalmazhat, hatékony megvalósításhoz szükséges tételekre, azok bizonyítására épülhet. Mivel minden fázis erős matematikai kidolgozottságot igényel, csak komoly elméleti ismeretekkel kezdhető el, s megértéséhez erőteljes absztrakciós készségre van szükség.

4. Feladattípus-orientált

Ebben az esetben az előző háromtól alapvetően eltérő módszerről van szó. Itt a programozás egységes tevékenység, egyes részei nem választhatók el egymástól, fontos jellemzője az előzőkkel szemben, hogy itt mindig a *teljes program*mal foglalkozunk. (Ez az összes további módszerre is igaz lesz.) Emiatt az egyes részterületeken váltakozva lép előre, s ismerünk meg újabb és újabb módszereket.

Itt egy konkrét *feladatkör*ből indulunk ki. Ez klasszikusan matematikai feladatkör, legtöbbször számelméleti feladatokkal (pl. oszthatóság, prímszámok, prímtényezős felbontás), de a módszer sikeres alkalmazása (elsősorban a közoktatásban) egészen más területekhez kötődik:

- grafika;
- szövegfeldolgozás;
- hétköznapi algoritmusok.

Mindegyik lényege az, hogy egy *egymásra épülő példák*at tartalmazó feladatsort kell megoldanunk. A feladatsor egyes feladatai megoldásához van szükségünk új programozási fogal-

makra, elemekre, s ezeket azért vezetjük be, mert a konkrét feladatmegoldáshoz kellenek. Ennek előnye, hogy az új ismeretet természetes igényekből kiindulva vezethetjük be, s nem kinyilatkoztatásként. Ugyanakkor azonnal alkalmazzuk is a feladat megoldására, s mint közismert, a megértés egyik legmagasabb szintje az, amikor az új ismeretet alkalmazni is tudjuk.

Különösen a hétköznapi algoritmusokra épített módszer alkalmas a programozással való kezdeti ismerkedésre (általános iskola alsó tagozat, sőt: óvoda), hiszen ez a mindenkiben meglévő természetes ismeretekre épít, abból vezeti le a programozási fogalmakat, módszereket. Fontos tudni: az óvodákban és az általános iskolákban hétköznapi algoritmusok megértését és végrehajtását már nagyon régóta tanítják, csak ennek nagyon sokáig semmi köze nem volt az informatika algoritmizálás tanításához. [K]

Megjegyezzük, hogy ezt az elképzelést (ti. az algoritmus úgy működik, ahogyan kézzel is csinálnánk) sok programozás-tanítási módszer alkalmazza.

5. Nyelvorientált

Ez az egyik legrégebbi módszer, az előzőhöz hasonlóan itt is mindig a működőképes program előállítása a megoldandó feladat. Másik fontos jellemzője, hogy valamely *programozási nyelv*hez kötődik, mint az alábbi két példából is látszik.

Egy tipikus párbeszéd részlete: – *„Te mit tanulsz programozásból?” – „Pascalt”.*

Ugyanez másképpen fogalmazva*:* Vannak *„Pascal”-programozók, „C”-programozók, …*

A módszer lényege, hogy egy *programozási nyelv*et tanít meg, s azon keresztül vezeti be a programozási ismereteket. Mivel a programozási nyelv van a középpontban, így sok nyelvfüggő ismeret is előkerül, sőt rögzül (!), mint általános programozási fogalom (gondoljunk például a BASIC nyelv DATA-READ-RESTORE utasításcsoportja használatához kapcsolódó ismeretekre). Emiatt az „egy *nyelven* tanult programozók” nagyon nehezen térnek át más programozási nyelvre.

A másik veszély abból származik, hogy az egyes programnyelvi elemek bonyolultságának szinte semmi köze a programozásbeli alkalmazásuk bonyolultságához, s emiatt a nyelvhez igazított tanításban nem megfelelő súllyal foglalkozunk vele. Kiváló példa erre az elágazás és az elöltesztelő feltételes ciklus utasítás (Pascalban: IF és WHILE), amelyek nyelvi szempontból körülbelül egyforma nehézségűek, a programozásban mégis sokkal többet foglalkozunk olyan feladatokkal, ahol a megoldásban ciklus szerepel, mint azokkal, ahol csak elágazás van.

Ugyancsak problémája a módszernek, hogy sok programozási fogalom, tevékenység (mint pl. verem, sor, rendezési módszerek, …) nem köthetők közvetlenül programnyelvi elemekhez (legalábbis ma még nem, ha az oktatásban számításba vehető programnyelvekre gondolunk), így a tanítási folyamatba beépítésük esetleges lehet.

6. Utasításorientált

Ez az előzőhöz hasonló, de nem egy konkrét programozási nyelvre, hanem egy *általános nyelvtípus*ra építő módszer. Lényegében csak ennyiben tér el tőle, azaz megmaradnak az álta-

lános nyelvhez kötöttség miatti problémák, s csak az egyetlen nyelv speciális ismeretei miatti gondok szűnnek meg.

A módszer definiálja az általános nyelvi elemeket, amelyek pl. egy Neumann-elvű felépítés alapján az alábbiak lehetnek:

- értékadás, kifejezések;
- beolvasás, kiírás;
- elágazások (kétfelé, sokfelé);
- ciklusok (számlálós, feltételes elöl-, illetve hátultesztelő);
- eljárások;
- függvények, operátorok;
- modulok.

A konkrét nyelvhez kötöttség problémáján kívül sajnos az előző módszer összes hátránya itt is megmarad, így ez a módszer is a veszélyesek közé sorolható.

7. Matematikaorientált

Ez az elképzelés egy másik tantárgy (példánkban éppen a matematika, de ugyanúgy lehetne más is) elképzeléseire épít. A megoldandó feladatsort a *matematikából* veszi, ahol az egyes feladatok a matematika szempontjából épülnek egymásra. Sajnos ez egyáltalán nem garantálja, hogy programozási szempontból is következetes a felépítés, valamint azt sem, hogy teljes.

Komoly csábítást maga a matematika jelenti „szuverén" témáival, sajátos belső logikájával, arányaival. Nem igazán garantálható, hogy ezen belső arányok szinkronba hozhatók a tényleges céllal, a programozással. Érdekes belelapozni ezen törekvés egy korai, egyébként igen színvonalas példáját jelentő könyvbe: Simonovits Miklós és Gémes Margit által írott tanári kézikönyvbe. [SG]

8. Hardverorientált

Elképzelése szerint az algoritmikus ismeretek nem érthetőek magas-szintű programnyelvi ismeretek nélkül; a programnyelvi ismeretek nem érthetőek assembly, ill. gépi kódú ismeretek nélkül; az assembly ismeretek nem érthetők a processzor működésének ismerete nélkül; ....

Emiatt a tetszetős, ámde téves következtetési lánc miatt a *programozási ismeretek*et megpróbálja *alulról felfelé* építeni. Állítja: „Aki ismeri a processzor működését, csak az érti, hogy miért éppen úgy épülnek fel az assembly utasítások, ahogy…, s hogyan kell egy assembly programot végrehajtani. Az assembly nyelv ismeretében már meg lehet fogalmazni, hogy hogyan működnek a magasszintű nyelvek utasításai. A magasszintű nyelvek utasításait ismerve könnyebb megérteni, hogyan működnek az egyes algoritmusok."

A módszer alapgondolatából következik, hogy a programozási (általánosabban: algoritmizálási és adatmodellezési) ismeretekre nincs szüksége mindenkinek, és az ilyen képzést elég az egyetemekre és főiskolákra hagyni. Ez pedig alapvetően ellentmond annak, hogy algoritmus megértésre, végrehajtási képességre mindenkinek (!) szüksége van, ez hangsúlyos része a közismereti informatikának.

Az egyes módszerek rövid értékelése

Úgy gondoljuk, hogy az első két módszer (algoritmus-, illetve adatorientált) a középiskolák vége felé (informatikus szakmákra készülőknek), az informatikai szakképzésben és a felsőoktatásban használható. Motivációs szempontból jobb az algoritmusorientált szemlélet, mivel ez tölthető meg sokféle, érdekesebb feladattal.

A specifikációorientált elképzelés csak egyetemek speciális informatikai szakjain (ún. elitképzésben) használható, ugyanis csak erős matematikai alapok megléte esetén lehet sikeres. Egyetemi „tömegképzésben" meggondolandó a 4-5. éves képzésben való megjelenése.

A feladatorientált módszer az, amelyet **egyetlenként** szabad alkalmazni a közoktatás minden olyan szintjén (általános és középiskolák), ahol elsősorban az algoritmikus gondolkodás kialakítása a fontos, és nem programozó szakemberek képzése.

A nyelv- és utasításorientált elképzelést elavultnak tartjuk, túlságosan sok veszéllyel járhat alkalmazásuk, hasznuk pedig lényegesen kisebb lehet más módszereknél.

A matematikaorientált módszert nem tartjuk célravezetőnek a programozás tanításában, megjegyezzük azonban, hogy programozási ismeretekkel rendelkezőknek nagyon hasznos lehet a programozással támogatott matematika-oktatás, hiszen mint említettük, az érti igazán a matematikát, aki alkalmazni tudja (például a programozásban).

A hardverorientált elképzelés (főleg végletes formájában) szintén elavultnak tekinthető, ez azonban sokkal érdekesebb kérdéseket is felvet: például, hogy jó helyen van-e az egyetemeken a számítógép architektúrákkal kapcsolatos tantárgy az első félévekben? Nem lenne-e értelme egyes részeit később, jóval mélyebben tanítani?

Zárszóként: mi az ELTE informatika tanárszakján a fenti gondolataink alapján a *Programozási módszertan*, valamint az *Adatszerkezetek és algoritmusok* tantárgy felépítésében az algoritmusorientált módszert választottuk. (Természetesen igazítjuk az egyes programozási lépések tárgyalását az egyetemen elvárható absztrakciós szinthez; így viszonylagosan sok formális ismeretet kell mozgósítaniuk az informatika tanárszakos hallgatóknak.) Ezen kívül több egyéb tantárgy (*Számítógépi grafika, Az informatika alkalmazás-módszertana*) is részben erre a koncepcióra épít. Tesszük ezt akkor is, ha tudjuk, hogy a nálunk végzett tanárok zöme nem ezzel a módszerrel fogja tanítani a programozást.

5. Irodalomjegyzék

[SzZs1] Szlávi Péter, Zsakó László: Módszeres programozás: Programozási tételek. ELTE TTK Informatikai Tanszékcsoport, 1996

[HSzZs] Harangozó É.-Szlávi P.-Zsakó L.: *Joining Programming Theorems, a Practical Approach to Program Building*. Annales Universitatis Scientiarum Budapestinensis. Sectio Computatorica 17, 155-172 , 1998

[PSzZs] Pap Gáborné, Szlávi Péter, Zsakó László: *Módszeres programozás: Adattípusok.* ELTE TTK Informatikai Tanszékcsoport, 1998

[SzZs2] Szlávi Péter, Zsakó László: *Módszeres programozás: Rekurzió.* ELTE TTK Informatikai Tanszékcsoport, 1995

[SzZsT] Szlávi Péter, Zsakó László, Temesvári Tibor: *Módszeres programozás: Programozás technológiája.* ELTE TTK Informatikai Tanszékcsoport, *1995*

[Zs] Zsakó László: *Módszeres programozás: Hatékonyság.* ELTE TTK Informatikai Tanszékcsoport, 1996

[SzZs3] Szlávi Péter, Zsakó László: *Módszeres programozás: Adatfeldolgozás.* ELTE TTK Informatikai Tanszékcsoport, 1995

[B] Bánné Varga Gabriella: *Programtervezési gyakorlatok*, SZÁMALK, 1989.

[F] Fóthi Ákos: *Bevezetés a programozáshoz.* Tankönyvkiadó, 1983.

[K] C.H.A. Koster: *Programozás felülnézetben*, Műszaki Könyvkiadó, 1988.

[SG] Simonovits Miklós, Gémes Margit: *Tanári segédkönyv Simonovits Miklós „Számítástechnika" tankönyvéhez,* OKKFT TS 4/1,1991