Programozás tanítási módszerek – stratégia a kezdetekre

Bernát Péter[1], Zsakó László[2]

[1]bernatp@inf.elte.hu, [2]zsako@caesar.elte.hu
ELTE IK

Absztrakt. A programozás tanítását alapvetően feladattípus-orientáltan képzeljük el [5], de nagyon sok függ a kiválasztandó feladattípusoktól. E cikkben azt járjuk végig, hogy az egyes választások milyen előnyökkel és hátrányokkal járhatnak. A következő stratégiákat nézzük végig: hétköznapi algoritmusok; teknőcgrafikára épített programozás, animációkészítésre épített programozás, játékfejlesztésre épített programozás, robotikára épített programozás; matematikára épített programozás.

Kulcsszavak: programozástanítás, feladattípus-orientált módszer, hétköznapi algoritmusok, teknőcgrafikára épített programozás, animációkészítésre épített programozás, játékfejlesztésre épített programozás, robotikára épített programozás; matematikára épített programozás

1. Bevezetés

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.

A 70-es évek végén más megközelítésben is előkerült a programozás tanításának módszertana a gyermeki gondolkodásra alapozva, ennek kialakítója elsősorban S. Papert [1]. Hasonló, bár más feladatkörre alapozott módszerről ír J. Hvorecky és J. Kelemen [2].

Koncepciójuk alapján a következők terjedtek el:

A. **Módszeres, algoritmusorientált** (a módszer az algoritmus megtervezését, megértését teszi a programozási folyamat középpontjába);
B. **Adatorientált** (a módszer az adatstruktúra megtervezését, megértését teszi a programozási folyamat középpontjába, ehhez alkot algoritmust, majd kódot);
C. **Specifikációorientált** (a módszer a specifikálást teszi a programozási folyamat középpontjába, s mindent ebből vezet le);
D. **Feladattípus-orientált** (a módszer egyre bővülő feladatsorokon keresztül vezeti be az új programozási fogalmakat, módszereket);
E. **Nyelvorientált** (a módszer egy programozási nyelv elemein keresztül tanítja meg a programozást);
F. **Utasításorientált** (a módszer egy általános programozási nyelv osztály elemein keresztül tanítja meg a programozást);
G. **Matematikaorientált** (a módszer egy másik tantárgy – pl. a matematika – lehetséges tananyagához vezet be programozási feladatokat, melyeket felfedezés-szerűen old meg);
H. **Hardverorientált** (a módszer alapgondolata, hogy programozási nyelv nélkül nem érthető a programozás, assembly nyelv nélkül nem érthető a programozási nyelv, … és így tovább egészen a számítógép működés fizikai alapjaiig);
I. **Mintapélda alapján** (a módszer kész programok elemzésén keresztül tanítja meg a programozást).

P. Szlávi és L. Zsakó szerint [5] a mindenkinek szóló informatika oktatásban a *D. Feladattípus-orientált* módszer az igazán jól használatos.

2. Feladattípus-orientált programozás tanítási módszer

A feladattípus-orientált programozás tanítási módszer sikeressége nagyban függ attól, hogy milyen feladattípusokat választunk, milyen stratégiával állunk hozzá a programozás tanításához.

A klasszikus programozás tanításában ez tipikusan matematikai feladatkör, legtöbbször számelméleti feladatokkal (pl. oszthatóság, prímszámok, prímtényezős felbontás).

Más stratégiát is választhatunk, de mindegyik lényege az, hogy egy egymásra épülő példákat tartalmazó feladatsort kell megoldanunk. A feladatsor egyes feladatai megoldásához van szükségünk új programozási fogalmakra, 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.

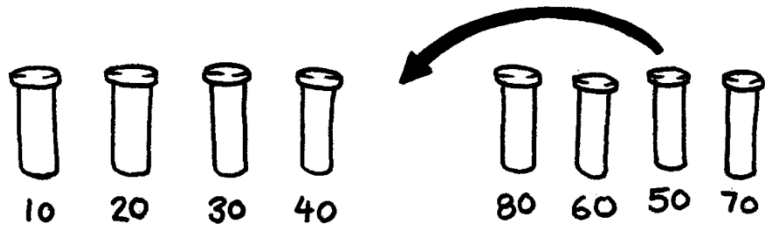
Ebben a cikkben a következő – feladattípus-orientált koncepcióra építő – stratégiákat fogjuk vizsgálni:

1. Hétköznapi algoritmusok
2. Teknőcgrafikára épített programozás tanítás
3. Animációra épített programozás tanítás
4. Játékfejlesztésre épített programozás tanítás
5. Robotikára épített programozás tanítás
6. Matematikára épített programozás tanítás

Az egyes stratégiákat elsősorban a következő szempontok szerint tárgyaljuk: a megtanítható programozási fogalmak és módszerek köre; kapcsolódási lehetőségek az informatika egyéb témaköreihez és más műveltségi területekhez; a továbblépés lehetőségei más nyelvek és az informatikus szakma irányába; a szociális kompetenciák fejlesztésének lehetősége; motivációs szempontok.

3. Hétköznapi algoritmusok

Mindennapi tevékenységeink során algoritmusokat hajtunk végre, amelyekben gyakran felfedezhetők az algoritmizálás és az adatmodellezés alapvető fogalmai és módszerei. Ezek az ismeretek megfelelően kiválasztott hétköznapi algoritmusokkal számítógép használat nélkül is bevezethetők.



1. ábra: Különböző súlyú skatulyák rendezése a minimum-kiválasztás módszerével [12]

Az óvodákban és az általános iskolákban hétköznapi algoritmusok megértését és végrehajtását már nagyon régóta tanítják, csak ennek sokáig semmi köze nem volt az informatika algoritmizálás tanításához [3].

Megjegyezzük, hogy azt az elképzelést, hogy az algoritmus úgy működik, ahogyan magunk is végrehajtanánk, sok programozás tanítási módszer alkalmazza.

A hétköznapi algoritmusokra építő stratégiával bevezethetők a következő, a mindennapi tevékenységeinkben is előforduló algoritmikus elemek és adatstruktúrák:

- szekvencia, elágazás, ciklus;
- eljárás;
- programozási tételek;
- elemi adatok, összetett adatok;
- sor, prioritási sor, lista, fa, gráf;
- objektum.

Nemigen találni példát azonban a változófogalomra és az értékadásra.

Különösen ez a stratégia támogatja a programozással való kezdeti ismerkedést (általános iskola alsó tagozata, sőt: óvoda), hiszen a mindenkiben meglévő természetes ismeretekre épít, azokból vezeti le a programozási fogalmakat, módszereket. [6]

Játékos, gyakran párban vagy csoportban végrehajtandó feladatok kapcsolódhatnak hozzá [12], amelyeknek nemcsak a megértésben és a begyakorlásban, de a tanulók szociális készségeinek fejlesztésében is szerepük lehet.

A módszer nagy előnye tehát, hogy természetes, hétköznapi tevékenységeken keresztül vezet be a programozás gondolatvilágába, de nagy problémája, hogy számítógép nélkül gyorsan unalmassá válik.

4. Teknőcgrafikára épített programozás tanítás

A teknőcgrafikát, mint a programozás tanításában jól használható módszert S. Papert [1] vezette be, már több évtizeddel ezelőtt. Újabban J. Hromkovic is hasonlókat fogalmaz meg [9].

A teknőc utasítható a számára „érthető” feladatok elvégzésére: adott távolsággal előre vagy hátra tud menni, adott szöggel jobbra vagy balra elfordulni, tollát (ami a hasára van erősítve) felemelni, leereszteni, más színűre cserélni, ezáltal mozgásával érdekes nyomokat hagyni a képernyőn. A teknőcgrafika fokozatosan egy új tudományterület, a teknőc-geometria megszületéséhez vezetett.

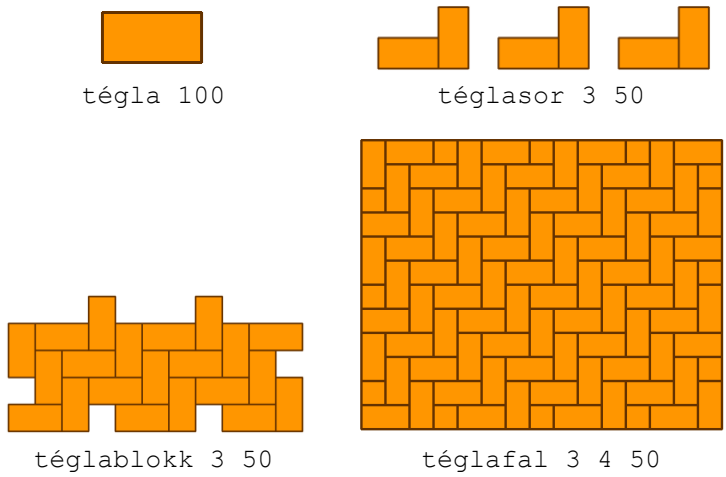
Alapgondolatai

- a programozás itt automataelvű – a végrehajtó teknőc egy automata, aminek helyébe saját magunkat is beleképzelhetjük;
- a specifikáció lényegi része egy rajz, a rajz elkészítése úgy történik, ahogy magam csinálnám.

Teknőcgrafika nagyon sok nyelvben van, így megvalósítása nem igényli speciális nyelv megtanulását, bár nyilvánvalóan a Logo programozási nyelvben vezették be, a nyelv erre a fogalomvilágra épül. A többi nyelvet általában ezzel a nyelvi eszközkészlettel bővítették.

A specifikáció itt pontos, betartása ellenőrizhető, hiszen a specifikáció maga egy rajz. Emiatt a feladatok látványosak, a szekvencia, az elágazás, a ciklus a rajzon látszik.

A teknőcgrafikában az eljárás a strukturálás alapeszköze, ahol az eljárások a rajzos specifikáció alapján megfogalmazhatók, mint részrajzok. (2. ábra)



2. ábra: Logo verseny mozaikos feladat

A rajz alapján a (paraméteres) eljárásokra bontás logikus, elvégezhető (még a rekurzió is), a paraméterek megjelenése következik a rajzos specifikációból. (3. ábra)

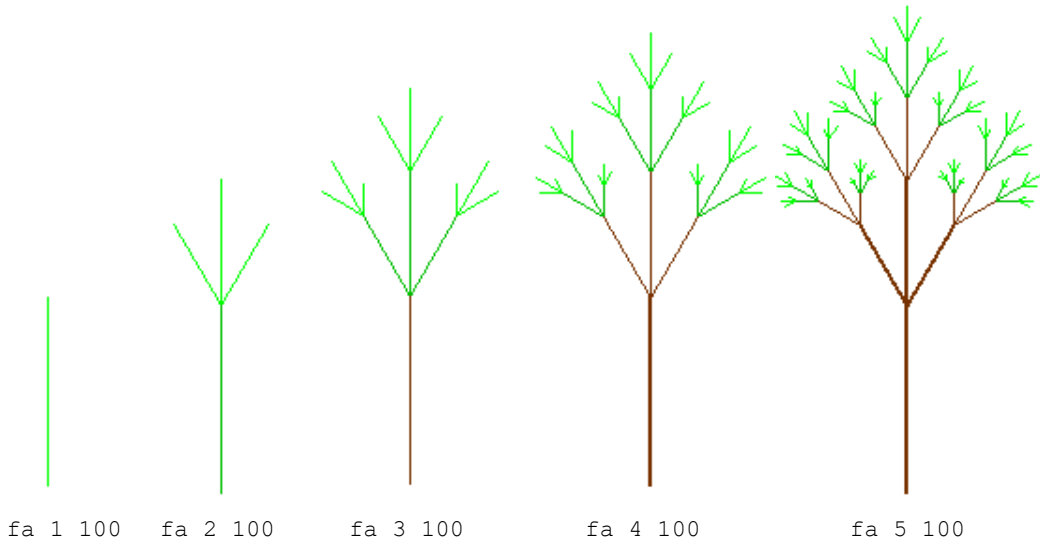
A témakör nagyon kevés nyelvi elemmel megvalósítható (Logo-ban: `előre`, `balra`, `tollszín!` a minimális utasításkészlet, de a teljes készlet sem túlságosan nagy). Emiatt a gondolkodást helyezi előtérbe a lexikális ismeretekkel szemben. A program nem utasítások hosszú lineáris sorozata, hanem – mint a szakmában általában – mélyen struktúrált szerkezet (azaz a sok gondolkodás mellett kevés gépelés kell hozzá). Alkalmas kevés tanulás után is „komoly" feladatok megoldására.

A rajzos programozás pozitív hatása – ami nem csak a teknőcgrafikára igaz -, hogy a hibás algoritmus lehet érdekes, a hibás rajz segítheti a hibakeresést. Akkor különösen jó ez a technológia, ha a felülről lefelé tervezett megoldásokat ki lehet alulról felfelé próbálni.

A megoldható feladatok miatt jól használható matematikai fogalmak tanítására, gyakorlására (a matematikát segítheti), kapcsolható koordinátageometriához.

A témakör jellegzetessége miatt sokféle feladattípus megvalósítható vele (pl. sokszögek, spirálok, sorminták, mozaikok, fraktálok, ásványszerkezetek, optikai csalódások, intarziák, mandalák, frízek stb.).

Továbblépési lehetőség a párhuzamos programozás felé: lehetőség több automatára (teknőcök). A színkezelés megoldása funkcionálisan, összetett típus kezelésével lehetséges, bevezethető vele a funkcionális programozás. A Logo alapú szimulációra remekül használható egy továbbfejlesztése, a netlogo.



3. ábra: Logo verseny fás feladat

Hátrányai közé tartozik, hogy logikusan nincs változó és értékadás, nincsenek típusok, nincsenek összetett típusok (csak, ami funkcionális nyelvekben lehet), továbbá pixelgrafikus feladatok megoldására nem jó.

Az objektumorientáltság ugyan megjelenik, de gyengén objektumorientált (teknőc).

5. Animációra épített programozás tanítás

Ennél a stratégiánál a programozási feladat egy animáció létrehozása. (4. ábra) Az animáció egyfajta forgatókönyv (mint specifikáció) alapján készülhet akár jelentős mennyiségű képi és hangi alapanyag felhasználásával, amelyek elkészítése is a feladathoz tartozhat. [7]

4. ábra: Egy jelenet az Alice programozási környezetben [7]

Animációk célszerűen objektumorientált programozással készíthetők. Ebben az esetben objektumok a szereplők, a díszlet elemei és az egyéb összetevők is (például kamerák és fényforrások). Az objektumok sokféle látványos tulajdonsággal és művelettel rendelkezhetnek, így segítségükkel szem-

léletes és közérthető módon vezethetők be az objektumorientált programozás alapfogalmai. (A kialakult képet azonban a későbbiekben nyilvánvalóan árnyalni kell, hiszen a programozás során használt objektumok az előzőeknél gyakran elvontabbak.)

Ebben a koncepcióban az eljárások a jelenetek, valamint a jelenetekben szereplők műveletei. Praktikus, ha a jelenetek önállóan is kipróbálhatók, a teljes film megalkotása nélkül. A jelenetek lineárisan követik egymást, ezért a program szekvenciális felépítésű. Nem jellemző a mély strukturálás, nincs rekurzió. Emiatt fennáll a „spagetti kód" alkotásának veszélye.

A szereplők tevékenységeit a jeleneteken belül folyamatosan össze kell hangolni, ami a pontos időzítésen kívül a szereplők közötti üzenetváltásokkal történhet.

Az új funkciókhoz sokszor új nyelvi elemek kellenek. Ez egyfelől nagy lexikális tudást igényel, ugyanakkor a sokféle különböző típusú (például a szereplők mozgatásával, a kinézetük megváltoztatásával vagy a hangokkal kapcsolatos) műveletnek köszönhetően a vezérlési szerkezeteket is látványosan eltérő célokra lehet használni. Így ezek a fogalmak általánosabban mutathatók be, mint például a teknőcgrafikában.

A stratégia jól támogatja a felülről lefelé kifejtés, illetve az alulról felfelé építkezés módszerét is: például bevezethetők jelenetek és objektumok a kifejtésük későbbre halasztásával, de önmagukban megvalósított jelenetek és objektumok is beilleszthetők a történetbe.

Az animáció készítéssel a kép- és hangrögzítő eszközök használatán, valamint a kép- és a hangszerkesztésen kívül könnyen kapcsolódhatunk bármely műveltségi területhez: megeleveníthetők történetek, készíthetők szemléltetések, bemutatók bármilyen témakörben.

Az objektumok elhelyezéséhez (ideértve a szereplőket és a kamerákat is) koordinátageometriai ismeretekre van szükség (amihez logikusan tartoznak változók és értékadás).

A feladatok motiválhatnak és gyors sikerélményt adhatnak, amelyek különösen a művészetekre (irodalom, filmművészet, zene) nyitott, és esetleg az elvontabb problémák iránt kevésbé érdeklődő tanulók számára lehetnek vonzók. Ezenkívül kellően összetettek lehetnek ahhoz (főleg, ha a képi és hangi alapanyagokat is el kell készíteni), hogy érdemes legyen rajtuk csapatban dolgozni. Kérdéses, hogy mennyire „lányos" az animációkészítés, történet kitalálás, illetve hogy mely korosztály fogja azt mondani, hogy unják már ezt.

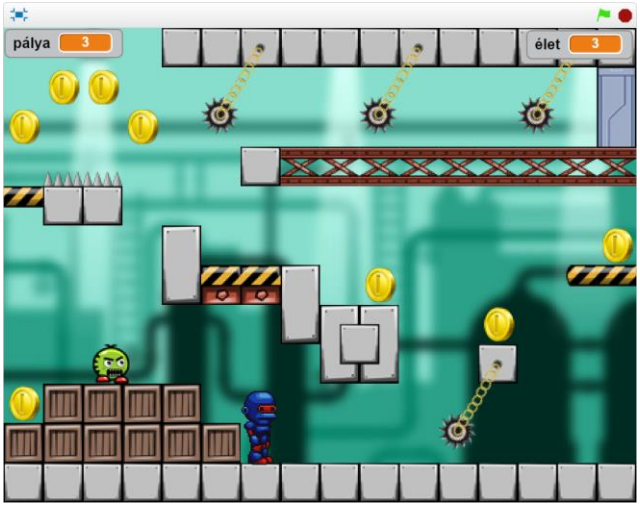
A stratégia problémái közé tartozik azonban, hogy bár az eredmény látványos lehet, de nem biztos, hogy megéri a látványosság adott szintje a befektetett munkát. Az igazán hatásos eredmény eléréséhez általában magas szintű (nem informatikai) ismeretekre van szükség. Következésképpen ez a stratégia az informatikai világból elvisz a filmkészítés világába, azaz nem sokáig segít az informatikus szakmára motiválásban.

6. Játékfejlesztésre épített programozás tanítás

A játék is bizonyos szempontból animáció, és az animáció készítésre alkalmas programozási nyelvekkel általában készíthetők játékok is. Az utóbbiak esetén azonban meghatározó a játékos és a játékprogram, továbbá a játékos és a számítógép által irányított szereplők közötti folyamatos interakció. Sőt, akár több játékos is játszhat ugyanazzal a játékkal, ami az online játékokra jellemző.

A játékfejlesztés tehát az animáció készítésre épül, ennek megfelelően az előző stratégia legtöbb előnye és hátránya itt is érvényes. Ebben a szakaszban elsősorban a különbségeket emeljük ki.

Mivel a számítógépes játékoknak a kezdetektől napjainkig nagyon sokféle műfaja alakult ki, könnyen meghatározhatók feladattípusok (például táblás- és kártyajátékok, autóversenyzős játékok, platformjátékok, építkezős játékok stb.) (5. ábra)



5. ábra: Egy platformjáték a Scratch programozási környezetben (saját játék)

A folyamatos interakciók miatt kiemelt szerephez jut az eseménykezelés. A játék és az egyes szereplők mindenkori állapotának nyilvántartásához pedig számos változóra és akár összetett adatszerkezetre is szükség lehet.

A játékfejlesztés módszerei az animáció készítésen alapulnak. Például a játékok is felbonthatók olyan jelenetekre, amelyekben a játék működése a többitől lényegesen eltérő (egy tipikus felbontás: főcím, menü, a játék futtatása, a játék vége). A játékkészítésnek azonban további játéktípus független, illetve játéktípus függő módszerei alakultak ki, amelyek ismerete szükséges az összetettebb játékok elkészítéséhez. Egyes játéktípusok kivitelezése csak jelentős matematikai és fizikai apparátussal lehetséges. Szükség lehet nagyon bonyolult, akár a mesterséges intelligencián alapuló viselkedésű szereplőkre.

A fentiek miatt a játékfejlesztés jó alapot és továbblépési lehetőséget biztosíthat az informatikus szakmák irányába.

A játékfejlesztés a számítógépes játékok megvalósítása iránt érdeklődők számára nagyon motiváló lehet. Mint ahogyan az is, hogy az elkészült játékokkal más is játszhat (például barátok). A játékkészítés esetén még indokoltabb lehet a feladat megosztása egy csoporton belül. Azt azonban figyelembe kell venni, hogy a szükséges ismeretek miatt a komolyabb játékok elkészítése a középiskolás korra, sőt annak is a végére tehető. Gy. Molnár és P Nyírő [11] egyetemi oktatásban próbálta ki a játékfejlesztésen alapuló programozás tanítási módszert, Game Maker-t használva.

Meggondolandó azonban, hogy a játékfejlesztés mennyire „fiús” tevékenység, segíti e a lányok nagyobb arányú megjelenését az informatikus szakmákban?

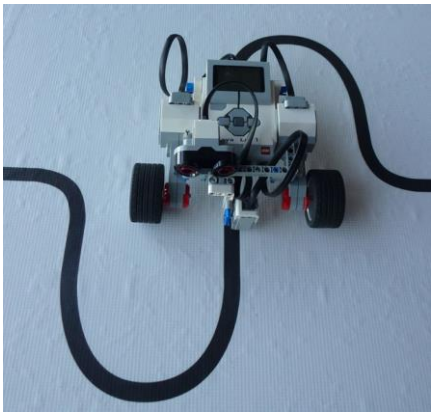
Ugyanennek a témának web-böngészőbeli megvalósításával foglalkozik Gy. Horváth, L. Menyhárt, L. Zsakó cikke [10].

7. Robotikára épített programozás tanítás

Ebben a stratégiában robotikai problémákat kell megoldani egy programozható robottal. Arra az alapgondolatra épít, hogy a fizikai eszközök programozása (az eszközök működésének megfigyelése) elősegíti az absztrakt programozási ismeretek kialakítását [8].

Az oktatási célú robotok jellemzően egyenesen haladni és kanyarodni képes szerkezetek, amelyekhez érzékelők is (elsősorban távolság-, ütközés-, fény- és színérzékelők) tartozhatnak. Az egyes

feladattípusokat az ilyen robotokkal megoldható problémakörök határozhatják meg (például területbejárás, átpakolás, útvonalkövetés, kijutás labirintusból stb.). (6. ábra)



6. ábra: Útvonalkövető LEGO robot [14]

A szóban forgó robotok néhány fajtája készre szerelt és a felépítésük nem módosítható, más típusok azonban a rendelkezésre álló elemekből mindig az aktuális problémának megfelelően építhetők össze. Szerelhető robot esetén a robotikai problémák szélesebb körével lehet foglalkozni, és a programozást jelentős mértékben kiegészítheti az elektromechanikai ismereteket igénylő robot- és pályaépítés. Nem szerelhető robot esetén a problémák szűkebb körével lehet foglalkozni, de a figyelem legnagyobb része a programozásra irányulhat.

Ezenkívül az oktatási robotok lehetnek valódiak vagy szimuláltak. A valódi robotok szó szerint kézzel foghatók és kapcsolatba kerülhetnek a valós világ tárgyaival, ezért működésük látványosabb. Egyszerre viszont csak kevesen tudják használni őket, és a program és a pálya módosítása körülményes lehet. Habár a szimulált robotok csak a virtuális térben léteznek, mindenki egyszerre használhatja őket, és a pálya és a program gyorsan változtatható és azonnal kipróbálható.

Programozási nyelvük a teknőcgrafikai programozási nyelvekhez hasonlóan automata elvű és eljárásorientált. A szenzorok állapotát jellemzően elágazásokkal lehet lekérdezni, de az eseménykezelés is elképzelhető. A változók és az adatszerkezetek használata azonban nem jellemző.

A robotikai problémák a tantárgyak közül elsősorban a matematikával és a fizikával lehetnek kapcsolatban. Bizonyos tanulók számára a fizikai világban mozgó robotok a többi stratégia virtuális tevékenységeinél motiválóbbak lehetnek. Ennél a stratégiánál is felmerül a párban vagy esetleg a kis csoportban dolgozás lehetősége: például a robot (és a pálya) megépítése, illetve a program elkészítése két külön részfeladat lehet.

A stratégiával kapcsolatban felmerülő probléma azonban, hogy az ipari robotok jelentős része nem mozog, a mozgó robotok pedig sokszor nem teknőcszerűen mozognak, hanem a mozgásállapotukat változtatják meg (pl. önvezető autók).

8. Matematikára épített programozás tanítás

Ez a stratégia a legrégibb, elsősorban a számelmélet témaköréből vett feladatokra épít, számelméleti feladatokat old meg (oszthatóság, prímszámok, számrendszerek), majd erre általában kombinatorikai feladatokat épít (faktoriális, Fibonacci számok, binomiális együtthatók). Újabban megjelentek a geometriai algoritmusok is.

Alapgondolata

- a matematikából ismert algoritmusok, módszerek megvalósítására koncentrál.

A matematika sok feladat algoritmusával, kiszámítási szabályával foglalkozik. Ezek sokasága alapvetően számtípusokra, valamint haladóbb szinten sorozatokra épít, amelyek megvalósítása nagyon könnyű bármely programozási nyelven.

A matematikai fogalmak használatából (osztó, prímszám, …) egyértelműen következik, hogy hogyan kell a programokat eljárásokra, függvényekre bontani.

Matematikai összefüggések alapján sokszor előkerülhet a rekurzió, mint a kiszámítás módszere, majd a rekurzió helyettesítése táblázatkitöltéssel (egyik tipikus példája a Fibonacci számok rekurzív összefüggése, majd táblázatkitöltéssel való megvalósítása.

Előnye és egyben hátránya is, hogy szorosan kapcsolódik a matematikához. Aki a matematikát nem szereti, azt ez a világ nem fogja meg. Aki viszont érti a matematikát, annak számára a matematikai algoritmusok megvalósítása nem okoz nehézséget, ezeken keresztül könnyedén megtanulja az algoritmizálási ismereteket.

Bár itt a „matematika" a címszó, természetesen más tantárgy is szóba jöhet, mint pl. P. Szlávi és L. Zsakó legelső tankönyvében [4], amely biológusoknak készült. Ebben olyan példák szerepelnek, hogy egy adott egyedről a genotípusa alapján állapítsuk meg, hogy heterozigóta-e, a vércsoportját megadó génpár szerint mondjuk meg, hogy A, B, AB vagy 0-s vércsoportú-e, két szülő vércsoportja alapján adjuk meg, hogy gyerekeik milyen vércsoportúak lehetnek, …

9. Záró gondolatok

Úgy gondoljuk, hogy nincs igazán üdvözítő módszer, amit kizárólagosan lehetne alkalmazni, minden esetben az előnyöket és hátrányokat mérlegelve kell dönteni.

Egyes módszerek kiválóan alkalmasak motiválásra, de a továbbhaladás velük nagyon nehézkes lehet. Más módszerek ezzel szemben a kezdeti lelkesedés után nagyon messzire is elvihetnek a programozás világában, a motiváció fenntartásával. Vannak módszerek, amelyek a gyerekek egyes rétegeinél nagyon sikeresek lehetnek, másoknál viszont csak kudarcba fulladhatnak.

Mindenképpen szem előtt kell tartanunk, hogy a programozás tanulás akkor lehet sikeres, ha érdekes és motiválhat az informatikus szakmákban való elmélyedésre, továbbá építkezhetünk rá a későbbi (akár szakmai) tanulmányok során.

Irodalom

1. Seymour Papert: *Mindstorms. Children, Computers and Powerful Ideas*, Basic Books, Inc, Harper Colophon Books, 1981.
2. Josef Hvorecky, Josef Kelemen: *Algoritmizácia, Elementárny Úvod*, Alfa, Bratislava, 1983.
3. C.H.A. Koster: *Systematisch Leren Programmeren*, Educaboek, 1984.
4. Péter Szlávi, László Zsakó: *Bevezetés a számítástechnikába.* Tankönyvkiadó, Budapest, 1987.
5. Péter Szlávi, László Zsakó: *Methods of Teaching Programming.* Teaching Mathematics and Computer Science, Vol.1. 2003.
6. Juraj Hromkovic: *Contributing to General Education by Teaching Informatics.* In: Mittermeir, R.T. (Ed.), ISSEP 2006, Lncs, 4226, 25–37.
7. Caitlin Kelleher, Randy Pausch: *Using Storytelling to Motivate Programming.* Communications of the Acm, July 2007/Vol. 50, No. 7., pp 59-64.

8. Attila Pásztor, Erika Török, Róbert Pap-Szigeti: *Innovatív informatikai eszközök és módszerek a programozás oktatásban.* Gradus, Vol.1., No.1, pp: 22-27, 2014.
9. Juraj Hromkovic: *Einführung in die Programmierung Mit Logo*, 2009, ISBN: 3834810045
10. Győző Horváth, László Menyhárt, László Zsakó: *Viewpoints of Programming Didactics at a Web Game Implementation.* XXIX. Didmattech 2016, Eötvös Loránd University, Faculty of Informatics, Budapest, 2016, pp. 79-88.
11. György Molnár, Péter Nyírő: *A gyakorlati programozás tanításának játékfejlesztésen alapuló, élménypedagógiai alapú módszerének bemutatása.* In: Karlovitz János Tibor: Pedagógiai és szakmódszertani tanulmányok, pp: 89-98, 2016.
12. Bell, T.; Witten, I. H.; Fellows, M.: *Computer Science Unplugged, an Enrichment and Extension Programme for Primary-Aged Children*; http://Csunplugged.Org/
13. *Alice – An Educational Software That Teaches Students Computer Programming in a 3d Environment*; http://www.Alice.Org/Index.Php/
14. *Drgraeme.Org – Free Tutorials Lego Ev3 Mindstorms*; http://Www.Drgraeme.Org