Algoritmizálás tanítása Flowgorithm-mel

Szabó Zsanett

szabo.zsanett@inf.elte.hu
ELTE IK

Absztrakt. Az algoritmizálás tanításának kérdése meglehetősen fontos és kényes. Amikor a 8. osztályos csoportjaimnál ez a témakör közeledett, akkor sokat gondolkoztam azon, hogy mivel és hogyan tanítsam ezt a témakört. A Flowgorithm programra esett végül a választásom. Így folyamatábrák segítségével ismertettem meg a tanulókkal az alapvető programozási szerkezeteket egy olyan környezetben, ahol lehetőség van az algoritmus futtatására, lépésenkénti futtatására, változókövetésre és 17 féle programozási nyelven megtekinthető, illetve futtatható fájlként letölthető az algoritmushoz tartozó forráskód is. A cikk a Flowgorithm program lehetőségeivel és az algoritmizálás témakör tanításakor történő használhatóságával foglalkozik.

Kulcsszavak: algoritmizálás, folyamatábra, Flowgorithm, módszertan

Bevezetés

Az algoritmizálás és a programozás tanításának kérdése meglehetősen fontos és kényes kérdés. Fontos, hogy ne vegyük el a tanulók kedvét a programozástól rögtön a legelején, ezért olyan felületre van szükség, ami barátságos, könnyen használható. Emellett az is kiemelt szempont lehet a tanítani kívánt program kiválasztásakor, hogy olyan tudásra tehessenek szert a tanulók a program segítségével, ami később más környezetekben is használható.

Amikor tavaly a 8. évfolyamos csoportjaimmal az algoritmizálás, programozás témakör felé közeledtünk, nagyon sokat gondolkoztam azon, hogy hogyan és mivel tanítsam meg nekik ezt a témakört úgy, hogy az számukra élvezhető legyen, de mégis később is hasznosítható tudásra tegyenek szert. Olyan felületet kerestem, amiben nem kell rögtön a legelején a szintaxissal bajlódni, de mégis több annál, hogy csak a dobozokat kell egymás mellé vagy alá pakolni, hiszen a tanítványaim egy része már találkozott ilyen felülettel. Elsődleges célom az volt, hogy az alapvető vezérlési szerkezeteket úgy tudjam megtanítani, hogy ne arra kelljen koncentrálniuk a diákoknak, hogy hogyan kell legépelni a kódot, hová kell pontosvessző vagy zárójel, hanem arra, hogy milyen logika áll a háttérben, mi mihez és hogyan kapcsolódik, mitől függ, hogy melyik kódrészlet mikor és hányszor fut le. Azt szerettem volna elérni, hogy később bárhol, bármilyen környezetben, bármilyen programozási nyelv esetén használt vegyék a tanultaknak. Az algoritmizálást és nem a kódolást szerettem volna előtérbe helyezni, mert azt gondolom, hogy később így könnyebben fognak majd boldogulni a bonyolultabb programozási feladatokkal is.

Devin D. Cook a Sacramentói Állami Egyetemen épp egy ilyen környezet létrehozását tűzte ki céljául. Olyan ingyenes felületet hozott létre 2014-ben, ami alkalmas az algoritmizálás és az alapvető vezérlési szerkezetek tanítására. A Flowgorithm lehetőséget ad a kezdő programozóknak arra, hogy először felesleges szintaktikai nehézségek nélkül, folyamatábrák segítségével ismerkedhessenek meg az alapvető programozási szerkezetekkel. A folyamatábrák használata nem új ötlet az algoritmizálás tanításánál, de a Flowgorithm több is egy folyamatábra-készítő programnál. A folyamatábrákat 17 népszerű programozási nyelven képes számunkra megjeleníteni, ráadásul úgy, hogy a folyamatábra és a programkód összetartozó részei azonos színű kiemelést kapnak, ami segíti a kezdő programozók számára a programkód megértését.

Algoritmus és tanításának fontossága

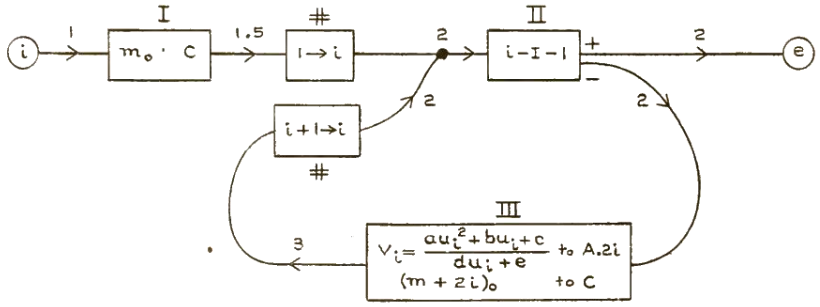
Az algoritmusok használata meglehetősen régre vezethető vissza, egészen a IX. századig. Az Arab Birodalomban Abbászida (Harun al-Rasid) uralkodása idején megépítették Bagdadban a Tudomány Házát, ahol a legnagyobb tudósokat gyűjtötték össze, hogy arabra fordítsák a görög műveket. Az egyik fordító felületes munkát végzett, mert al-Hvárizmi arab szerző aritmetikai jellegű, a hindu tízes számrendszert ismertető, eredetileg *„Ezt mondja al-Hvárizmi:…”* kezdetű címmel rendelkező könyvének címében fordításkor a szerző nevét *Algorithmi*-ra változtatta. Innen származik az algoritmus elnevezés, ami több nyelven is elterjedt [1, 2].

Az informatika tantárgy algoritmizálás ismeretkörének célja elsősorban az algoritmikus gondolkodás kompetencia fejlesztése [3]. Azaz foglalkozni kell benne algoritmusok megértésével, végrehajtásával, tervezésével és megvalósításával. Ezek jól használhatóak a modellezés, a problémamegoldás, valamint a rendszerszintű gondolkodás fejlesztésére is. Ezen az ismeretkör tanításakor, tanulásakor a hétköznapi algoritmusok világától fokozatosan jutunk el a programozási ismeretek egy meghatározott köréhez. A számítógépes algoritmusoknál a világ dolgait adatokkal írjuk le, az adatokból egy program kiszámít más adatokat, amelyeket megjelenítve megismerhetjük a valós világot. Az egyes korosztályok ismereteinek olyannak kell lennie, hogy arra a következők építhessenek, azt ne kelljen megcáfolniuk. Így például, mivel a végcél strukturált algoritmusok készítése, ezért már a kezdet kezdetén is ehhez kell illeszkedniük a megfogalmazott algoritmusoknak.

A folyamatábrák történetéről

Az első folyamatábrák kialakulásának története [4] szerint 1921-ben kezdődött, amikor Frank és Lillian Gilbreth bemutatta a folyamatfejlődés dokumentációjának első strukturált metódusát, a „folyamatfejlődés-diagramot” az American Society of Mechanical Engineers (ASME) tagjainak. Gilbreth-ék eszköze hamar bekerült a mérnöki tantervekbe. Az 1930-as évek elején egy ipari mérnök, Allan H. Mogensen kezdett el olyan eszközöket tanítani üzletembereknek egy New York-i konferencián, amiket a mérnökök használtak. Mogensen tanítványai is továbbfejlesztették az eszközt. Egyikük, Ben S. Graham információfeldolgozáshoz alakította át folyamatfejlődés-diagramot. Az ő fejlesztésével többszörös folyamatokat és azok kapcsolatait is ábrázolni lehetett. 1947-ben az ASME elfogadott egy szimbólumkészletet, amit Glibreth-ék eredeti munkája alapján alakítottak ki.

Douglas Hartree 1949-ben elmagyarázta, hogy Herman Goldstine és Neumann János korábban kifejlesztettek egy folyamatábrát (eredetileg diagramot) számítógépes programok tervezéséhez. Hartree korabeli jelentőségét igazolják az IBM mérnökei és Goldstine személyes visszaemlékezései. Goldstine és Neumann eredeti folyamatábrái egy nem publikált beszámolójukban [5] tekinthetők meg, illetve az **1. ábra** is egy a beszámolóban szereplő folyamatábrák közül.



1. ábra: Folyamatábra Goldstine és Neumann beszámolójából [5]

A folyamatábrák a számítógépes algoritmusok leírásának népszerű eszközévé váltak. Népszerűségük az 1970-es években csökkent, amikor az interaktív számítógép-terminálok és a harmadik generációs programozási nyelek váltak a számítógép-programozás népszerű eszközévé. Manapság arra használnak folyamatábrákat, hogy számítógépes algoritmusokat írjanak le velük. Az UML tevékenységi diagramokról és Drakon ábrákról vélik úgy, hogy azok a folyamatábra kiterjesztései.

A ma használatos folyamatábrák a programot gráfként írják le, ahol a programgráf egy irányított gráf, amely csomópontokból és az azokat összekötő élekből áll. Egyetlen induló és befejező éle van, az induló élből bármely csomópont elérhető, s bármely csomópontból el lehet jutni a befejező élre [6].

Algoritmizálás tanítása folyamatábrákkal

Az algoritmizálás tanításának egyik közkedvelt módja annak folyamatábrákkal történő tanítása. Több ismerőstől hallottam már, hogy régen ő is folyamatábrákkal tanulta az algoritmizálást, bár akkor még papíron készítették el azokat. Van, ahol kifejezetten a folyamatábrák használatát ajánlják kezdők számára [1], illetve több helyen összegyűjtötték a folyamatábrák használatának előnyeit, illetve hátrányait [1, 6, 7, 8].

Az előnyök közül azt gondolom, hogy az egyik legnyilvánvalóbb az, hogy látványos, szemléletes. Így kívülállók számára is könnyen érthető, áttekinthető. A folyamatábrákban nem kell a strukturált vezérlési elemeket használnunk, csak az elkészült, „nyers" folyamatábrát kell később átalakítanunk a strukturált vezérlési szerkezeteknek megfelelően. A folyamatábrák lehetőséget adnak számunkra arra, hogy a valódi folyamatot írjuk le, és úgy készítsük el az algoritmust, ahogyan eszünkbe jut. Bizonyos hibákat már tervezés közben kiszűrhetünk, amikor elkészítjük a folyamatábrát, illetve segít abban, hogy programozás közben „ne tévedjünk el", ne bonyolódjunk bele a végrehajtási folyamatokba.

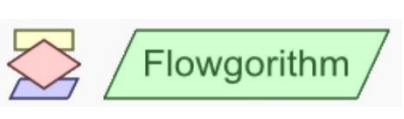
Mint mindennek, természetesen a folyamatábrák használatának is vannak hátrányai is. Az egyik ezek közül éppen az, hogy nem támogatja a strukturált programozást, vagyis nincs külön szimbólum például a ciklusokra, illetve az algoritmus folyamatábrájában gyakorlatilag bárhová mutathatnak a nyilak. Másik hátránya lehet még az, hogy összetettebb feladatok esetén igen terjedelmessé és kuszává válhat a folyamatábra, de úgy gondolom, hogy összetettebb feladatok esetén gyakorlatilag elkerülhetetlen az, hogy a feladathoz tartozó algoritmus is összetetté, bonyolultabbá váljon. Megfelelő strukturálás és logikus felépítés mellett nem mondanám azt, hogy a folyamatábra értelmezése nehezebb lenne, mint bármilyen más módon készített algoritmus értelmezése összetett feladatok esetén.

Problémaként merül fel, hogy a folyamatábrák szövegszerkesztővel nehezen szerkeszthetők, javításuk nehézkes, a papír alapú megoldás választásakor pedig nem tudhatjuk előre, hogy mekkora papírlapra lesz szükségünk. Szerencsére különböző folyamatábra-készítő programok segítségével ma már ez sem jelent akkora hátrányt. Találtam olyan cikket [9], amiben több ingyenes eszközről is írnak, amivel lehet folyamatábrákat készíteni.

A folyamatábrákkal kapcsolatos alapprobléma pedig [6] szerint az, hogy strukturális alapelemei nem azonosak a szokásos algoritmikus szerkezetekkel, sőt segítségükkel más struktúrák is létrehozhatók. A Flowgorithm segítségével készített folyamatábrák esetén szerencsére ez a probléma nem áll fenn, mert csak előre meghatározott elemek közül választhatunk, és azokat csak *szabályosan* helyezhetjük el. Az elkészített folyamatábra pedig minden esetben átalakítható futtatható programkóddá.

A Flowgorithm

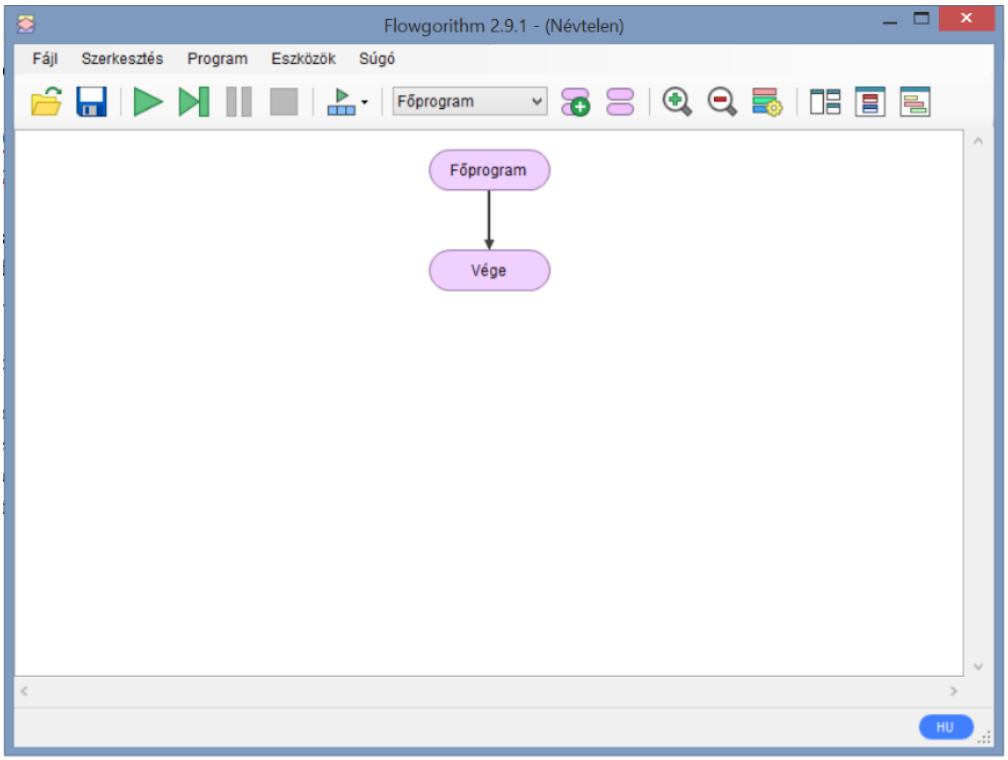
A Flowgorithm elnevezésű ingyenes program neve két szó, a *flowchart* (folyamatábra) és az *algorithm* (algoritmus) összevonásából jött létre. A program egy olyan programozási



környezetet biztosít számunkra, amiben folyamatábra formájában készíthetjük el programjaink algoritmusát. Az elkészített algoritmust futtathatjuk egészében vagy lépésenként, futtatás közben figyelhetjük a változók értékeit és ráadásként még többféle programozási nyelven is megtekinthetjük az algoritmushoz tartozó kódot.

A Flowgorithm programot folyamatosan fejlesztik. Első változata 2014-ben jelent meg, jelenleg pedig a 2.9.1-as a legfrissebb verzió, amit 2017. november 4-én tettek közzé. A program nagy előnye, hogy huszonhat nyelven, köztük magyarul is elérhető és csak kis (4 Mb-nál kevesebb) helyet foglal. A Flowgorithm egyelőre csak Windows-os környezetben használható, de ígérik, hogy később Linux-ra és Macintosh-ra is telepíthető lesz.

A program felülete rendkívül egyszerű, könnyen átlátható és kezelhető. Megnyitásakor rögtön egy *üres* program algoritmusát láthatjuk (**2. ábra**), ami egy főprogram és egy vége dobozból áll. Nincs zavaróan sok menü, így használata, lehetőségei egyszerűen megtanulhatók. Számomra egyedül a visszavonási lehetőség hiányzik a programból. A grafikus ikonok beszédesek, jelentésük könnyen elsajátítható. Számos színezési beállítás közül választhatunk, illetve a „dobozok" formáit is többféle szabvány szerint állíthatjuk be.

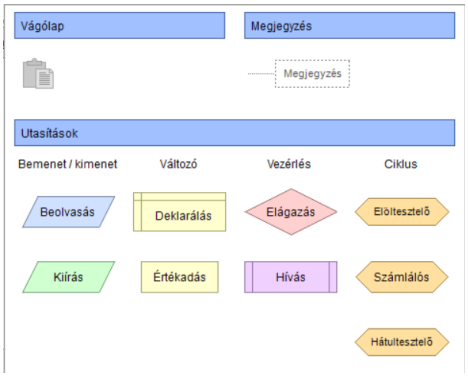


2. ábra: A Flowgorithm megnyitáskor

A program használatának bemutatásához és a kezdeti lépéseinek megkönnyítésére nyolc lépésből álló kezdő tutorial érhető el a program weboldalán [10]. Ez egy *Hello world* program elkészítését mutatja be programképek és rövid magyarázó szövegek segítségével. A magyarázó szövegek angol nyelvűek, de a képek garantálják a megértést az angol nyelvet nem értők számára is.

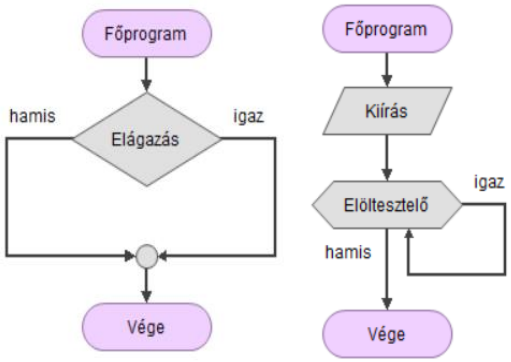
A *Hello world* tutorial mellett tizenkét példaprogramot is közzétettek a Flowgorithm weboldalán [10]. Ezek letölthetőek, megtekinthetőek, kipróbálhatóak, szerkeszthetőek, így segítik a kezdő felhasználókat a program adta lehetőségek megismerésében. A programhoz súgó is elérhető, de csak angolul és olaszul. Várható, hogy a későbbiekben ez még változni fog, mert a honlapon többek között magyar fordítókat is keresnek.

A program megnyitásakor létrejövő *üres* programban két dobozt látunk, a főprogram és a vége dobozokat. Az őket összekötő nyílra kattintva van lehetőségünk további dobozok beszúrására. Itt az alábbi lehetőségek közül választhatunk:



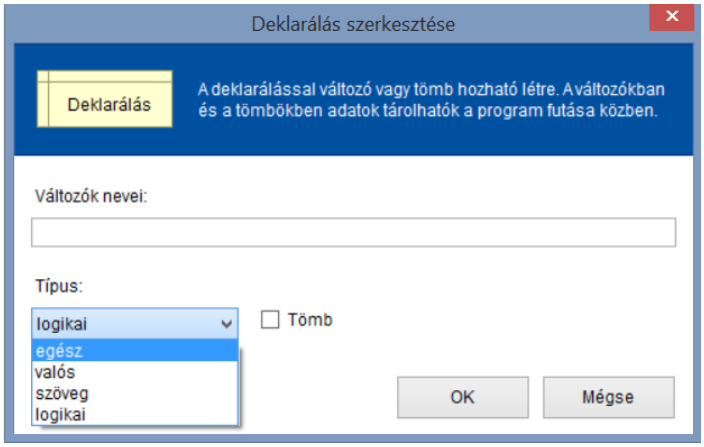
- bemenet/kimenet:
  - beolvasás: bekérhetünk adatokat a felhasználótól
  - kiírás: kiírhatunk a képernyőre
- változó:
  - deklarálás: változókat hozhatunk létre
  - értékadás: változók értékeit állíthatjuk be, módosíthatjuk
- vezérlés:
  - elágazás: kétirányú elágazást hozhatunk létre
  - hívás: eljárásokat, függvényeket hívhatunk meg
- ciklus: a megfelelő ciklusfajtát szúrhatjuk be
  - elöltesztelő
  - számlálós
  - hátultesztelő
- megjegyzés: megjegyzést szúrhatunk be
- vágólap: beilleszthetjük a korábban vágólapra helyezett elemeket

Ezek a lehetőségek beszédesek, megfelelnek a szakkifejezéseknek. Minimális ismeretek elsajátítása után könnyű kitalálni, hogy melyik mire való. A szakkifejezések nem ismerése viszont nem jelent hatalmas hátrányt a kezdők számára, mert a lehetőségek kiválasztása után a megjelenő nyilak és dobozok (**3. ábra**) alapján érthetővé válhat a szakkifejezés is.



3. ábra: Példa a megjelenő üres dobozokra és nyilakra

Változó deklarálása esetén egész, valós, szöveg és logikai változók közül választhatunk a Deklarálás szerkesztése ablakban (**4. ábra**), illetve ezekből tömböt is létrehozhatunk. A logikai változók értéke *true* és a *false* lehet. Ezek a kifejezések nincsenek magyarra fordítva, de ez a két angol szó nem okozhat nagy nehézséget a fiatalok többségének.

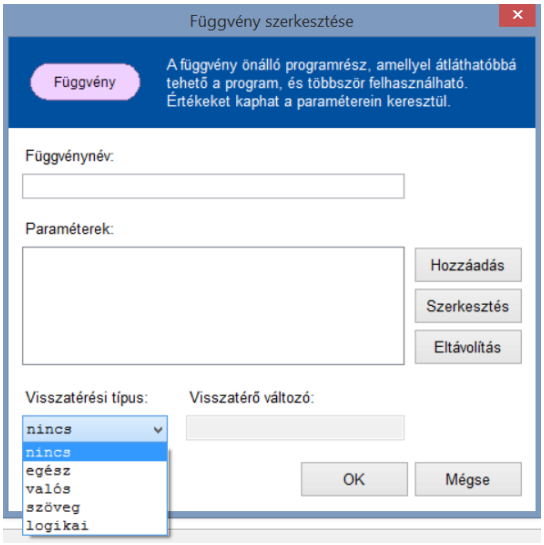


4. ábra: Deklarálás szerkesztése ablak

A program beépített függvényekkel is rendelkezik:

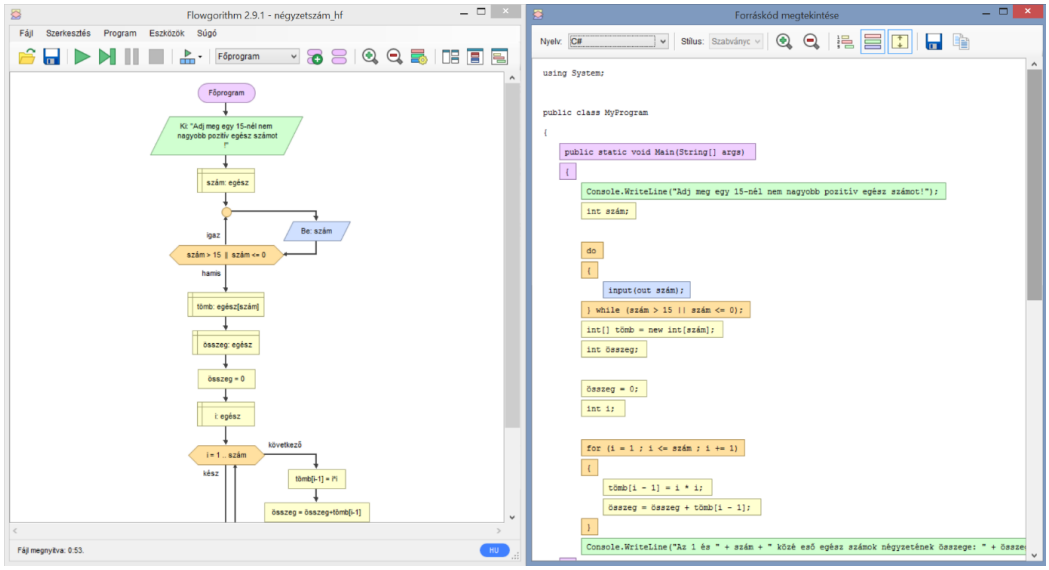
- **Matematikai:** `Abs(n), Arcsin(n), Arccos(n), Arctan(n), Cos(n), Int(n), Log(n), Log10(n), Sgn(n), Sin(n), Sqrt(n), Tan(n)`
- **Szöveg:** `Len(s), Char(s, i)`
- **Konverziós:** `ToChar(n), ToCode(c), ToFixed(r, i), ToInteger(n), ToReal(n), ToString(n)`
- **Egyéb:** `Random(n), Size(a)`

Ezek mellett pedig saját eljárásokat, függvényeket is létrehozhatunk, melyek visszatérési érték nélküliek vagy egész, valós, szöveg, illetve logikai visszatérési értékkel rendelkezhetnek. Erre a függvény hozzáadása gomb segítségével van lehetőségünk a Függvény szerkesztése ablakban (**5. ábra**). A meglévő függvényeket pedig a függvénykezelőben kezelhetjük.



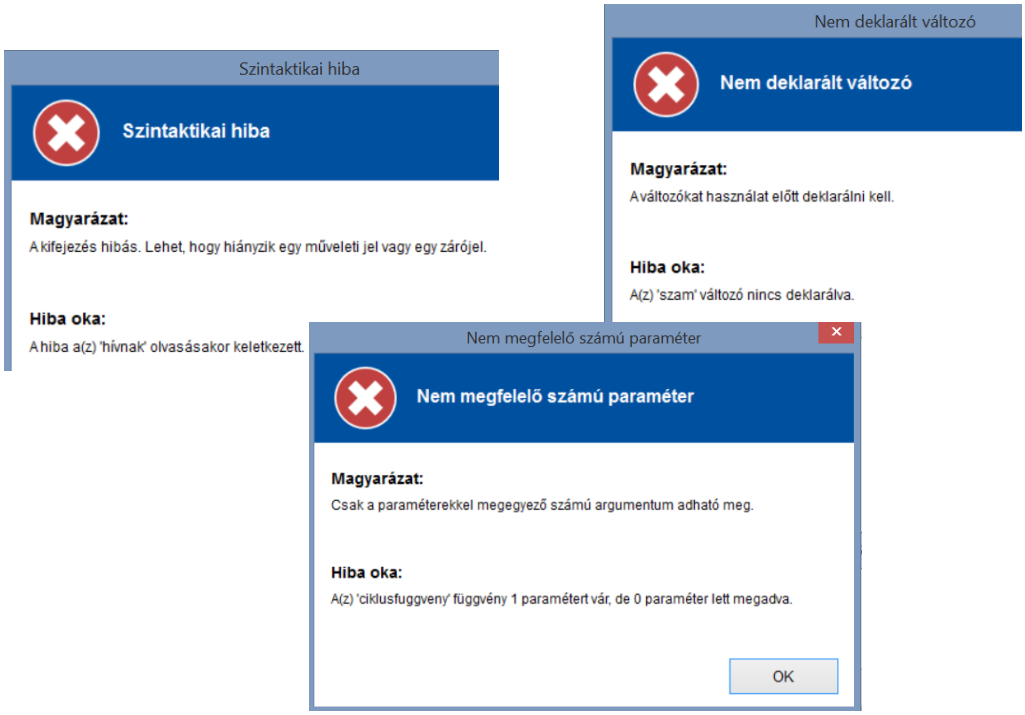
5. ábra: Függvény szerkesztése ablak

A létrehozott algoritmus kódját tizenhét programozási nyelven megtekinthetjük: C#, C++, Delphi/Pascal, IBO Pseudocode, Java, JavaScript, Lua, Perl, Python, QBasic, Ruby, Swift2, Swift3, VBA, Visual Basic .NET, Gaddis Pseudocode. Ezekbe a kódokba (az algoritmussá történő visszafordítás nehézségei miatt) nem lehet beleírni, viszont az így keletkezett kód letölthető a kiválaszott nyelvnek megfelelő formátumban és úgy már szerkeszthetővé válik. A kód megjeleníthető úgy, hogy annak egyes sorai az algoritmus dobozainak megfelelő háttérszínt kapják (**6. ábra**). Ez nagy segítséget jelenthet a kezdő programozóknak, akik még csak ismerkednek a programozási nyelv szintaxisával. Hiszen a színek segítségével egyértelművé válik számukra, hogy a kód melyik része tartozik az algoritmus egyes lépéseihez.



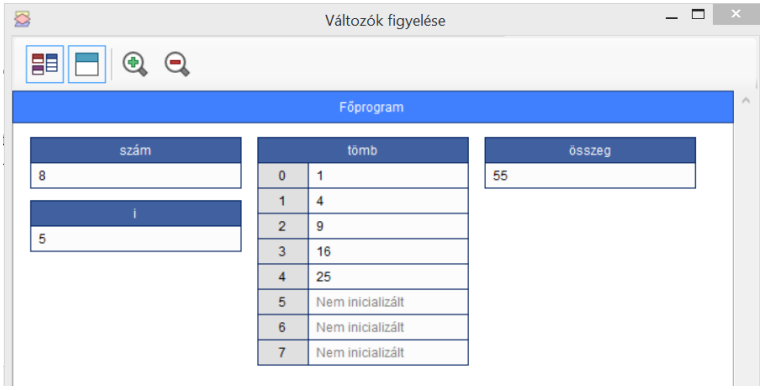
6. ábra: Azonosan színezett algoritmus és forráskód

A magyar nyelvűre állított program magyar nyelvű hibaüzeneteket küld, ami nagy könnyebbséget jelenthet a kezdő felhasználóknak. A hibaüzenetek nemcsak hogy magyar nyelvűek, hanem egyértelműek is (**7. ábra**).



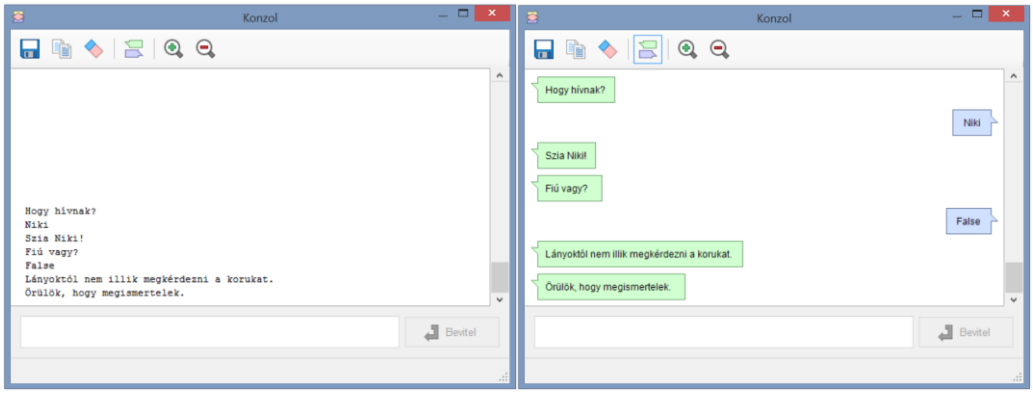
7. ábra: Néhány hibaüzenet

Az elkészített algoritmust egyben és lépésenként is futtathatjuk. A futtatás során a változókövetésére is lehetőség van (**8. ábra**). Ez nagy segítséget jelent a vezérlési szerkezetek tanításakor, főként a ciklusok megértésénél.



8. ábra: Változók figyelése ablak futtatás közben

A konzol ablakban kétféle megjelenítés közül választhatunk (**9. ábra**). Az egyik esetben a többi programozási környezet konzol ablakára jobban hasonlító ablakot kapunk, míg a másik esetben egy barátságos, párbeszédes ablak segítségével követhetjük az eseményeket.



9. ábra: Konzol ablak kétféle megjelenítése

A Flowgorithm program segítségével a legegyszerűbb programok algoritmusaitól kezdve jól felépíthetők az összetettebb programokhoz szükséges algoritmusok. A diákok későbbi tanulmányaik során építhetnek az így megszerzett tudásra, hiszen a megszerzett ismeretek jól illeszkednek a később elsajátítandó ismeretekhez, nem lesz szükség később a Flowgorithm segítségével tanultak cáfolására.

A Flowgorithm-mel kapcsolatos tanítási tapasztalatok

A Flowgorithm tanítási szempontból új programnak mondható, hiszen három éve jelent meg, így több éves tanítási tapasztalatokról és oktatásban való használatának hosszútávú hatásáról nem igazán beszélhetünk. A program kialakításából eredően azonban sejthető, hogy tanítási célra szánták. Erről sajnos nincsenek pontos információim, mert a szerzőnek írt levelemre nem kaptam választ. Az egyik népszerű közösségi oldalon viszont van egy csoport, ahol a program iránt érdeklődő felhasználók meg tudják osztani egymással a gondolataikat. Ebben a csoportban érdeklődtem azután, hogy kinek milyen tanítási tapasztalata van a Flowgorithm használatához kapcsolódóan. Túl sok információhoz sajnos ilyen módos sem sikerült hozzájutni, de ketten leírták a tapasztalataikat.

Paul Shaddick (Bristol) 16-18 éveseknek tanít programozást C# programozási nyelven. Tanítványai között többen soha nem programoztak korábban. Nekik a Flowgorithm különösen sokat segít a szekvencia és az iteráció megértésében.

Jamie Cropley (tanuló) pedig más tanulóknak szokta javasolni a program használatát. Azt írta, hogy problémásnak találja a program használatát akkor, ha az ember a mesterséges intelligencia irányába szeretne elmenni. A Flowgorithm használatát kifejezetten kezdőknél találja jónak. Ő maga soha nem használt folyamatábrákat egyszerű feladatok megoldásához, de tudja, hogy dokumentáció készítésekor nagyon jól jön. Annak megértéséhez, hogy mit és hogyan kell csinálni nagyon jó eszköznek tartja a programot, hiszen a későbbiekben jól lehet építeni a megszerzett ismeretekre.

Saját tanítási tapasztalat

Az algoritmizálás témakörével összesen 11 tanórán tudtunk foglalkozni a tavalyi tanév végén. Heti egy alkalommal volt óránk, ezért gyakorlásként minden óra után kisebb házi feladatot kaptak a tanulók. Az elsődleges célom az volt, hogy az alapvető vezérlési szerkezeteket sajátítsák el a diákok úgy, hogy megértik a működésüket is.

Három párhuzamos 8. osztályos csoporttal dolgoztam egy hatosztályos gimnáziumban. A tanulók kevesebb mint a fele (42%-a) tanult korábban valamiféle programozást. Ők főként Logo-val vagy Scratch-csel foglalkoztak, néhányan Arduino-val, Java-val, illetve HTML-lel és CSS-sel.

Az első bevezető órán egyszerű, hétköznapi algoritmusok írásával kezdtünk. A tanulók egyénileg készítették el viszonylag kevés instrukció alapján az italautomata használatának lépéseit, majd közösen egyre jobban finomítottuk, további lépésekre bontottuk a meglévő lépéseket. A tapasztalatok alapján és azokat kiegészítve megfogalmaztuk, hogy mi az az algoritmus és mit várunk el az algoritmusoktól.

A második órától kezdve sorra megismerkedtünk a fontosabb műveletekkel, vezérlési szerkezetekkel. Az algoritmusok készítése közben rendszeresen megjegyzésekkel tűzdeltük tele a munkánkat, hogy ne csak a kész algoritmus maradjon meg, hanem legyen nyoma az oda vezető gondolatmenetnek is. Először a beolvasással, kiírással, a logikai változóval, elágazással és az & jel kiírásban történő használatával ismerkedtünk meg. Az algoritmizálás témakört táblázatkezelés előzte meg, ami abból a szempontból szerencsés volt, hogy több dologra is vissza tudtam utalni. Például az " " és & jel szerepére, illetve a HA függvényre.

A 3., 4. és 5. órákon a háromféle ciklussal ismerkedtünk meg (hátultesztelős, számlálós, elöltesztelős sorrendben). A 6-7. órákon a % művelettel, egy-egy szám osztóinak meghatározásával és a csoport matematikai tudásszintjének megfelelően hatékony prímszám-e algoritmust készítettünk. A 8-10. órákon az eljárások, függvények és véletlen számok generálása került elő egy lottósorsolásos feladat kapcsán. A 11. órán pedig szövegfüggvényekkel kapcsolatos feladatokkal dolgoztunk.

Törekedtem arra, hogy a tanulók számára az látszódjon, hogy egy-egy felmerülő probléma megoldásához van szükség az újabb és újabb szerkezetek megismerésére. Nem szerettem volna, hogy erőltetettnek tűnjenek a feladatok, amiket csak azért kell megoldani, hogy megtanulhassuk az összes szerkezetet. Nagy örömömre sokszor a diákok is javasoltak feladatokat, módosítási lehetőségeket a meglévő feladatokhoz, ráadásul olyanokat, amik alkalmasak voltak arra, hogy dolgozzunk velük. Sikerült megvalósítani azt, hogy együtt gondolkozott a csoport, egymás gondolatait egészítették ki és tökéletesítették. Számomra úgy tűnt, hogy a többség határozottan élvezte az órákat, de még a témakör iránt kevésbé érdeklődő tanulókat is sikerült elgondolkodtatni az órákon.

Számonkéréskor nem kellett önállóan algoritmust készíteniük a diákoknak, mert úgy gondoltam, hogy azzal nem tudnám megfelelően számonkérni mindazt, amit szeretnék. Ezért két része volt a számonkérésnek. Az első részében egy tesztet kellett kitölteniük a tanulóknak, amiben például igaz/hamis állítások voltak a ciklusokkal kapcsolatosan, volt, ahol meg kellett mondani, hogy hányszor fut le egy ciklus, illetve volt, ahol össze kellett párosítani a vezérlési szerkezeteket a neveikkel. A dolgozat második felében pedig egy kész algoritmus esetén kellett leírni, hogy mi látszódna a változók figyelése ablakban, illetve meg kellett fogalmazni, hogy milyen feladatot old meg az algoritmus.

Összességében elégedett voltam azzal, amit az órákon és a dolgozatok javításakor tapasztaltam, hiszen úgy éreztem, hogy sikerült elérni azt, amit célul tűztem ki.

Tanulók visszajelzései

A saját tapasztalataim jók voltak, de kíváncsi voltam arra is, hogy a diákok mit gondoltak a programról, mi tetszett nekik az órákon és mi nem. A tanulók visszajelzései összességében pozitívak voltak. Néhány példa arra, hogy mi tetszett nekik:

- *„Jól átlátható volt a színezés és az alakzatok miatt."*
- *„A különböző színű ablakoknak különböző funkciójuk volt."*
- *„Egyszerű felület."*
- *„Viszonylag könnyen megértettem vele mindent. Megtanultam vele programozni."*
- *„Objektív, jó hogy panelekből épül fel a programsor, logikus."*
- *„Az, hogy nagyon egyszerűen meg lehetett érteni."*

- *„Az, hogy kisebb feladatokat tudtunk elvégezni vele, aminek látható eredménye volt."*
- *„Könnyen el lehet sajátítani vele a programozás alapjait (magát a programozási nyelv logikáját)."*

A diákok a programból többségében nem hiányoltak semmit. Néhányan a programhoz tartozó részletes leírást, használati útmutatót hiányolták (valójában jogosan), ami valóban csak angolul és olaszul lenne elérhető jelenleg. Ahhoz, hogy teljes legyen a kép, ahhoz azt is le kell írnom, hogy akik korábban már többféle programozási nyelvvel is találkoztak, szívesen láttak volna nagyobb kihívásokat is, például külső fájlok importálását. Erre jelenleg nem alkalmas a program, de talán nem is kellene, hogy az legyen, hiszen ez már nem tartozik hozzá azokhoz az alapokhoz, ami a program létrejöttét indokolhatta.

Tudni szerettem volna, hogy szívesen ajánlanák-e másoknak is a Flowgorithm programot, akik nem hozzájuk hasonlóan nem tanultak még programozni vagy csak nagyon keveset, de szeretnének később programozással foglalkozni. A tanulók több mint 70%-a ajánlaná másoknak.

- *„Azért ajánlanám, mert lépésenként meg lehet tanulni a programozását."*
- *„Mert érdekes és jó bevezető a programozásba."*
- *„Alapvetően követhető és érthető, ha valakinek jól elmagyarázzák, akkor meg fogja érteni."*

Azok, akik nem ajánlanák másoknak, többségében azt írták, hogy azért, mert tanári magyarázat és segítség nélkül nem értették volna, hogy hogyan kell megoldani a feladatokat. Úgy vélem, hogy már a 70%-os többség is elég meggyőző volt, de ha hozzávesszük azokat a tanulókat is, akik csak tanár nélkül nem ajánlanák a programmal való ismerkedést, akkor kb. a diákok 85%-a ajánlaná a Flowgorithm programot az alapok elsajátításához.

Összegzés

A Flowgorithm programot a tapasztalataim alapján szívesen ajánlom érdeklődő csoportok számára, akik nem tanultak még programozni (vagy csak nagyon minimálisan) és nyitottak arra, hogy valami újat tanuljanak. Az én csoportjaim 8. osztályosok voltak, de jól megválasztott feladatok mellett alacsonyabb évfolyamoknál is beválhat a program használata.

Hivatkozások

[1] P. Mohos, 2008. szeptember 19. http://www.uni-obuda.hu/users/mohosp/2008_9_2/Algoritmus/Algoritm_.htm. (utoljára megtekintve: 2017.11.08.).

[2] F. László: A tudományok királynője: a matematika fejlődése, Typotex Kft., 1997.

[3] P. Szlávi, L. Zsakó: *Informatika oktatása*, http://tamop412.elte.hu/tananyagok/infokt/lecke5_lap1.html. (utoljára megtekintve: 2017.11.08.).

[4] *Wikipédia – Flowchart*, https://en.wikipedia.org/wiki/Flowchart. (utoljára megtekintve: 2017.11.08.)

[5] H. H. Goldstine, J. v. Neumann: *Planning and coding of problems for an electronic computing instrument* (Part II, Volume 1), in Institue for Advanced Study Princeton, New Jersey, 1947.

[6] T. Gregorics, V. Heizlerné Bakonyi, G. Horváth, L. Menyhárt, G. S. Pap, Z. Papp-Varga, P. Szlávi és L. Zsakó: *Programozási alapismeretek*, http://progalap.elte.hu/downloads/seged/eTananyag/lecke4_lap1.html. (utoljára megtekintve: 2017.11.08.)

[7] A. Göllei: *Folyamatmérnöki ismeretek alapjai (tantárgytematika és vázlat nem lektorált változat)*, Szeged, 2011.

[8] T. Markó, 2003. nostromo.pte.hu/~alex/FEEK/3felev/Adatbaziskezeles/03_algoritmus.ppt. (utoljára megtekintve: 2017.11.08.)

[9] T. Horváth: *ArtCharacter*, 25 05 2012. http://www.artcharacter.hu/friss/software/22-useful-free-tools-for-creating-charts-diagrams-and-flowcharts/. (utoljára megtekintve: 2017.11.08.)

[10] *Flowgorithm – tutorial*, http://www.flowgorithm.org/documentation/tutorial-8.htm. (utoljára megtekintve: 2017.11.08.).

[11] *Flowgorithm*, http://www.flowgorithm.org/. (utoljára megtekintve: 2017.11.08.).

[12] *Flowgorithm – example programs*, http://www.flowgorithm.org/documentation/index.htm. (utoljára megtekintve: 2017.11.08.)