Király Sándor
Eszterházy Károly Főiskola
ksanyi@aries.ektf.hu

TEHETSÉGGONDOZÁS AZ INFORMATIKÁBAN CSÖKKENŐ ÓRASZÁMOK MELLETT

Abstract

Az új NAT az informatikai műveltségterületet elvileg kiemelten kezeli. A kerettantervek szerint azonban jelentősen csökkent az informatika órák száma a középiskolákban.[1] Egyes szakemberek szerint ennyi órában még a NAT követelményeket sem lehet teljesíteni, még akkor sem, ha az integrált oktatás keretein belül is szerepet kap az informatika. Ha még az érettségire történő felkészítésre sem elég az óraszám, akkor hogyan lehet majd ezek után a nagy informatikai versenyekre (Nemes Tihamér, OKTV) felkészíteni a diákokat, hiszen a követelmények ezeken a versenyeken már eddig is messze meghaladták az informatika érettségi követelményszintjét? A megoldást éppen az informatika adja. Megfelelő IKT eszközök alkalmazásával, motiválással a helyzet akkor sem reménytelen, ha már egyetlen informatika órája sincs a diáknak. Ebben cikkben szeretnénk megmutatni, hogyan lehet még akkor is sikeresen felkészíteni diákokat az informatika OKTV-re, Nemes Tihamér versenyre, ha közben már nulla óraszámban tanulnak az iskola falain belül informatikát, vagy ha tanulnak is heti egy órában, nem azt, amire a versenyen szükségük van.

Bevezetés

Ma Magyarországon az informatika versenyek közül a legrangosabb az OKTV és annak „belépő versenye” a Nemes Tihamér Országos Informatikai Tanulmányi Verseny (NTOITV). Az előbbi versenyen elért 1-30. helyezés többletpontokat biztosít a tanulóknak, az utóbbin való részvétel segíti a felkészülést az OKTV-re. Mindkét versenyen alkalmazói és programozói kategóriában lehet indulni. Ennek megfelelően egy középiskolai diáknak, ha informatikából versenyezni akar, akkor ezeken a versenyeken érdemes megmérettetnie magát, tanárának pedig segítenie kell a felkészülését. Ha megnézzük a verseny követelményeit, akkor kiderül, hogy azok még az emelt szintű érettségiét is messze meghaladják, sőt, programozásból olyan algoritmusokat is ismerniük kell a versenyzőknek, melyeket már a felsőoktatásban tanítanak.[2] Heti egy órában, de még kettőben sem tűnik egyszerű vállalkozásnak még a tehetséges és elkötelezett diákok megfelelő felkészítése sem. Sok iskolában pedig éppen az utolsó két évben, amikor a diákok az OKTV-n indulhatnak, már egyáltalán nem tanítanak informatikát. Ebben a cikkben megmutatjuk, milyen módszerek, eszközök segíthetnek abban, hogy a tanulók ilyen óraszámok mellett is eredményesen szerepelhessenek a két versenyen.

Felkészítés az alkalmazói versenyekre

Az NTOITV alkalmazói kategóriájának követelményei: rajzolás és képszerkesztés, szövegszerkesztés, táblázatkezelés, prezentációkészítés és weblapszerkesztés. Az OKTV-n ezeken kívül még adatbázis-kezelés tudásukról kell a versenyzőknek számot adniuk.

A kilencedikes diákok már általános iskolában is tanulnak képszerkesztést és szövegszerkesztést, ennek megfelelően az eddig ismeretlen szövegszerkesztési és képszerkesztési műveletek megmutatására elegendő 5-6 óra. Ezután következhet a gyakorlás. Ezekre a versenyekre célszerű úgy felkészíteni a tanulókat, hogy a korábbi évek feladatait próbálják megoldani. Ha ezek elfogytak, jöhetnek a tanár saját feladatai. Bár minden diák ugyanazt a feladatok fogja kapni, nem fogja tudni mindenki megoldani, csak a tehetségesebbek és az elkötelezettebbek. Ők újabb feladatokat kapnak, melyeket nem csak órán, hanem otthon is megpróbálnak megoldani, a többieknek órán lehet segíteni, még egyszer megmutatni, mit hogyan kell. A gyengébbeknek célszerű könnyebb feladatokat adni. A potenciális versenyzők közben haladhatnak, gyakorolhatnak tovább.

A további feladatokat megkaphatják e-mail-ben csatolt állományokban, vagy akár egy ingyenes tárhelyen megoszthatjuk velük a feladatokat. Mivel nem egyformán haladnak, célszerű versenyzőként külön mappát megosztania a tanárnak. A megoldás kerülhet ide, vagy kaphatja a tanár e-mailben is. A megoldások ellenőrzésére nincs automatizálási lehetőség, a kapott dokumentumokat, képeket végig kell nézni a javító kulcs alapján.

Mivel az első fordulóban táblázatkezelési feladatok is szerepelnek (bár kisebb súllyal), néhány hét után ezzel a résszel is foglalkozni kell a felkészítés során. Az egyszerűbb függvényeket már ismerik az általános iskolából, a többi, versenyen szükségeseket azonban meg kell mutatni. A felkészülést segítheti a korábbi versenyeken használt függvények összegyűjtése, példák segítségével történő bemutatása egy dokumentumban. Ezek nagy részét így önállóan, otthon is megtanulhatják a tanulók megfelelő tananyag esetén. A gyakorlás pedig ismét elsősorban otthonra marad. A második és harmadik fordulóra történő felkészítés a prezentációkészítés és a weblapkészítés elsajátítását is megköveteli a diáktól. Így, ha nincs külön óra, szakkör számukra, csak a kötelező órák, akkor a többi diákkal együtt kell ezeket megtanítani. Csak miközben a nem versenyzők 1-2 feladatot tudnak megoldani, addig a versenyzők otthon folyamatosan tudnak készülni, a megoldásokat elküldve vagy feltöltve a tanár folyamatosan tudja ellenőrizni a munkájukat. Órán, miközben a nem versenyzők gyakorolnak, meg lehet beszélni a versenyzőkkel a hibákat.

A fentiek szerint folytatva az oktatást, az adatbázis-kezelést kivéve a teljes gimnázium anyag megtanítható a versenyezni akaró tanuló számára. Ennek megfelelően az első év végére informatikai tudásban teljesen heterogén társaságot kap a versenyeztető tanár még akkor is, ha nincs külön órája, szakköre a felkészítésre.

Egy informatikát szerető diák nyáron úgy pihen, hogy versenyfeladatokat old meg, hiszen a többi tárgyat ilyenkor nem kell tanulnia. Ezt a bő két hónapot kihasználva hatalmas lépésekkel lehet haladni a cél felé, a versenyzők egyre magabiztosabban, egyre gyorsabban tudják megoldani a feladatokat egy jól sikerült nyári felkészítés után. Ilyenkor érdemes a táblázatkezelési ismeretekre fókuszálni, hiszen a versenyeken ez a

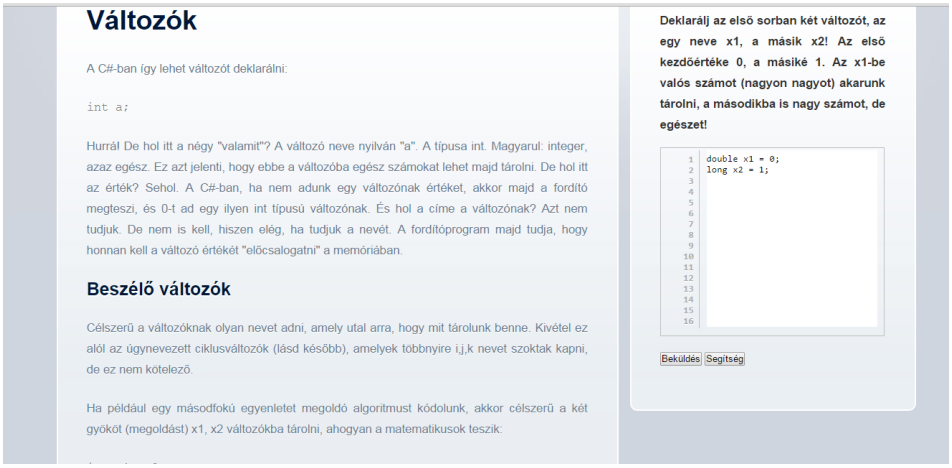
vízválasztó. Aki ebben megfelelő ismeretekkel, jártassággal rendelkezik az akár az első tízben is végezhet, egyébként nincs esélye a jobb szereplésre.

Felkészítés a programozói versenyekre

Általános iskolákban többnyire nem tanítanak magas szintű programozási nyelveken programozni, jobb esetben algoritmusok leíró eszközökkel, valamint a LOGO nyelvvel találkozik a diák. A középiskolában a programozás tanítása sokkal nehezebb, hiszen egyrészt többnyire nincs megfelelő előképzettség, ráadásul a tanítási órákon az alkalmazói ismereteket kell tanítani. Ugyanakkor az algoritmusok megértése, a kiválasztott programozási nyelv szintaktikájának és szemantikájának elsajátítás is sokkal nagyobb nehézségbe ütközik, mint egy szövegszerkesztő vagy egy prezentáció-készítő eszköz használatának, alkalmazásának elsajátítása. A középszintű érettségin nem követelmény a programozás, emelt szinten pedig az alapvető programozási tételek felhasználásával elkészíthető program kódolása a követelmény, miközben már az NTOITV második fordulójában is többet kell tudnia egy olyan diáknak, aki a döntőbe szeretne jutni. Az OKTV-n és a Nemzetközi Informatikai Diákolimpia (IOI) válogatóversenyeken pedig már a gráf bejárási algoritmusok és alkalmazásaik, geometriai algoritmusok, minimális feszítőfa megkeresése stb. alapkövetelmény, ha valaki például 2015-ben Kazahsztánban szeretne az olimpián részt venni.

Kezdeti lépések, interaktív, kódolást segítő tananyag

A hagyományos tanítás során a diákoknak két tankönyvre, jegyzetre van szükségük a felkészülés elején. Egy olyanra, amelyben az alapvető algoritmusok találhatók és egy olyanra, amelyben a kiválasztott magas szintű programozási nyelv elemei kerültek bemutatásra. A tanár pedig az órán ismerteti, magyarázza ezeket, a tanuló pedig otthon a tankönyvekre támaszkodva tudja az anyagot megtanulni. Ezzel a módszerrel még heti 1 órában is igen nehéz megtanítani egy jó képességű diákot programozni.



1. ábra: Új tananyag és kódolás egy időben

A legtöbb idő a nyelv elemeinek megismerésére, használatára, azaz a kódolásra megy el. Ennek megismerésére, begyakorlására, egyszerűbb algoritmusok kódolására egy olyan programot érdemes írni, melyet a diák otthonról, webes felületen elérhet, és interaktív módon tanulhatja meg a nyelv elemeinek használatát, egyszerűbb algoritmusok kódolását.

Az általunk készített tananyag PHP nyelven[3] íródott, egy Linux operációs rendszeren futó szerverre (*kodolosuli.nejanet.hu*) lett telepítve, a tananyagok, a feladatokat és az egyes diákok eredményeit MySQL adatbázisban tároljuk. Az adott nyelv megtanulásához szükséges tananyag oldalakra tördelve a baloldalon található. A jobboldali panel tetején az elméleti részhez kapcsolódó feladatokat láthatja a tanuló. A feladatok megoldását, azaz a kódokat a jobboldali panelbe kell a diákoknak begépelni. Ha befejezték a kódolást, alul a „Beküldés" gombra kell kattintaniuk. Ha a kód helyes, akkor a következő leckét jeleníti meg a program. Ha helytelen, akkor hibaüzenet kapnak, utalva a hiba sorára, jellegére, és tovább kell próbálkozniuk. A Segítség gombra kattintva a program megpróbál olyan instrukciókat adni, amelyből a megoldás kitalálható. Ha nem sikerül a helyes kódot megadni a feladatra, akkor is tovább lehet lépni a következő leckére. A tanulói próbálkozásokat adatbázisban tároljuk, így a tanár által nyomon követhető az egyes tanulók próbálkozásai. A főoldalon látható, hogy mely leckéket teljesítette a diák, és melyeket nem.

A cél az, hogy a diák **önállóan** megtanuljon olyan szinten kódolni a kiválasztott nyelven, hogy az alapvető programozási tételeket (keresés, kiválogatás, szélsőérték keresés stb.) le tudja programozni. Így tanórán már a fejlesztői környezetet használva gyorsabban lehet feladatokat megoldani, a kódolást gyakorolni, és az algoritmusokra, azok megértésre koncentrálni.

Programkiértékelő rendszer a gyakorlás támogatására

A programozási tételek megismerése, gyakorlása után a tanítás következő lépése a gyakorlás, az alapalgoritmusokra, majd az egyre nehezebb algoritmusokra épülő feladatok megoldása. A megoldások után szükség van az elkészített programok tesztelésére, értékelésre. Valóban minden inputra helyes eredményt ad-e a tanuló programja? A további, nehezebb algoritmusokra épülő feladatok megoldása során még kritikusabb a programok kiértékelése.

Az sajnos fel sem merül, hogy órán oldjanak meg a diákok feladatokat, majd ezeket még az órán ellenőrizzük, megbeszéljük. Ez ennyi órában lehetetlen. Ennek megfelelően a diák az általa otthon elkészített, helyesnek vélt programot elküldi a tanárának, aki teszteli a diák programját tesztadatai segítségével. A diák pedig majd kap egy választ, hogy jó vagy hibás volt-e a programja. Utóbbi esetben a tesztadatok ismeretében javíthatja azt, majd a folyamat kezdődhet előről. Szerencsésebb esetben 1-2 nap, rosszabb esetben több nap múlva derül ki, hogy a feladatot sikerült-e tökéletesen megoldani.

A felkészüléshez a legjobb módszer a korábbi feladatok, illetve azokhoz hasonló feladatok megoldása. [4] Egy Nemes Tihamér vagy egy OKTV győzelemhez, egy IOI-ra való kijutáshoz nagyon sok feladatot kell még egy jó képességű diáknak is megoldania, végigjárva a fenti utat. Persze az is előfordulhat, hogy a diák a feladattal együtt megkapja a tesztadatokat is, és nem kell várnia a kiértékelésre. Ez viszont szakmailag

nem túl előnyös, hiszen a versenyeken sem állnak rendelkezésre a programozás során az értékeléshez szükséges tesztadatok.

A döntőben már nem csak meg kell oldani egy problémát, hanem a lehető legjobban kell megoldani, hiszen a feladatok itt már idő és memória korlátokat is tartalmaznak. A maximális pontszám eléréséhez az optimális megoldást kell megtalálnia a programozónak, ez a pontszámot azonban csak akkor kaphatja meg, ha a limiteket nem lépi túl. Ezért tartjuk rendkívül fontosnak, hogy a diákok ne csak a döntőben találkozzanak ilyen rendszerrel, hanem a felkészülés során is. Az on-line értékelők használatával a tanulók nagy része sajnos csak a verseny találkozik először.

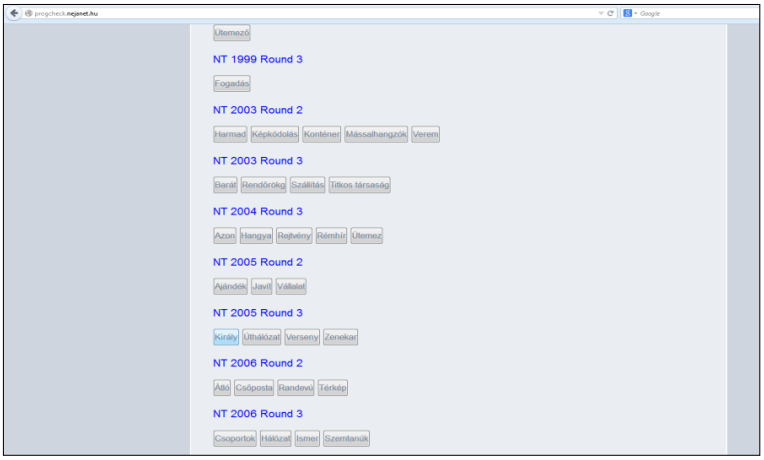
Ezekre a programokra az a jellemző, hogy a verseny előtt a diákok hozzáférést kapnak rendszerhez, amelybe feltölthetik elkészített forrásnyelvű programjaikat. A webes felülettel rendelkező program a kiértékelés után minden teszteset esetén közli a versenyzővel, hogy a programja helyes vagy helytelen megoldást adott-e, illetve időlimites-e a programja, rosszabb esetben futási hibával állt-e le, valamint az elért pontszámot. Ennek ismeretében a versenyző dönthet, hogy megpróbál javítani a kódon vagy egy újabb feladatba kezd. A verseny végén a tanuló pontosan tudja a pontszámát, így a verseny végeredménye is hamar ismertté válhat.

A rendszerek azon az elven működnek, hogy a versenyző programját lefordítják, és különböző tesztadatokra lefuttatják. Ez eredményül kapott output fájl tartalma alapján eldöntik, hogy a program helyes eredmény adott-e, illetve hány pontot ér a megoldás. [5]

Egy ilyen programot használva nemcsak az értékelést tudjuk automatizálni, hanem a felkészítés is gyorsabbá válhat, kevesebb órára lesz szüksége a tanárnak a felkészítés során. Egy ilyen programkiértékelő rendszer írása egyáltalán nem nehéz feladat egy középiskolai tanár számára sem, hiszen programozni minden tanár tud, így már csak egy megfelelő szerver szükséges a megvalósításhoz.

A szerző által Ruby nyelven megírt, a Ruby on Rails keretrendszer felhasználásával készített szoftver az iskolai Linux szerveren (*progcheck.nejanet.hu*) fut. [6][7] A rendszer adatbázisa tartja nyilván a tanulókat, versenyzőket, a feladatokat és azok kategóriáit, a mappaszerkezete pedig a feltöltött programokat, tesztállományokat és a kiértékelő programokat. A feltöltött programok sandbox környezetben futnak, alapértelmezett idő és memória limitet kapnak, melyek feladatonként változtathatók. A rendszer használói kategóriákba vannak sorolva, nem látja mindenki az összes feladatot, csak a rendszer adminisztrátora. Minden feladathoz input és output, valamint a pontszámokat tartalmazó állományok tartoznak, melyek felhasználásával az alapértelmezett értékelő program dönt a program helyességéről és az elért pontszámokról. Minden tesztesethez külön pontszámok rendelhetők. Ha egy input esetén több megoldás is lehetséges, tetszőleges nyelven megírt értékelőt kell feltöltenie az adminisztrátornak a helyes értékeléshez az alapértelmezett értékelő helyett.

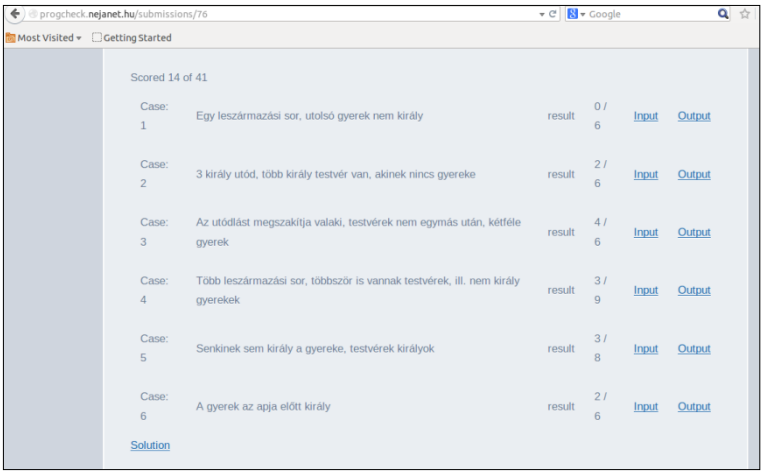
A feltöltött programnak Java, C++, C, C# nyelven kell íródnia, a Java esetén a main() függvényt tartalmazó osztálynak kötelezően a main nevet kell kapnia. A felkészítésben résztvevő tanulók hozzáférést kapnak a rendszerhez, melyben az elmúlt 15 év Nemes Tihamér, OKTV, IOI válogatóverseny, IOI, CEOI feladatainak egy része található, de tetszőleges feladatok megoldásainak kiértékelésére lehetőséget nyújt.



2. ábra: A kiválasztott feladat megoldásának feltöltése

A tanulók a bejelentkezés után a megoldott feladat nevének ismeretében feltöltik megoldásaikat, majd a kiértékelés után megkapják, hogy melyik teszteset esetében értek el pontot, és melyiknél nem. Ha a fordítás sikeres, akkor helyes, helytelen, idő túllépés, futási hiba válaszokat kapnak. Az első esetet kivéve a diák megkapja az input és az output állományt is, és azt, hogy ezekkel a fájlokkal mit kívántunk tesztelni. A tesztadatok ismeretében a diák önállóan is tudja javítani programját.

A feladatokhoz (maximális pontszám esetén is) egy javasolt megoldást is mellékel a rendszer PDF formátumban. Ezek ismeretében a diák önállóan is képes javítani programját, illetve akár újra is írhatja. Ha mégsem, még mindig rendelkezésre áll a tanári segítség. Az eddig tapasztalatok alapján erre csak akkor volt szükség, ha a megoldási javaslat nem volt világos a diák számára.



3. ábra: A beküldött feladat kiértékelése

A rendszer admin felületébe bejelentkezve a tanár láthatja, hogy melyik diák melyik feladatot milyen sikerességgel oldotta meg, hány beküldés után jutott el a végleges, hibátlan megoldáshoz. Mivel minden beküldés eredménye megőrzésre kerül, így a mappaszerkezet ismeretében az adminos rész nélkül is láthatóak az eredmények, megtekinthetők a beküldött programok forráskódja. Ennek megfelelően a tanár bármikor tanulmányozhatja a kódokat, adhat refaktorálási tanácsokat a diák számára.

Beállítható, hogy a beküldésről a tanár e-mailt kapjon, az e-mailhez csatolt állományban láthatja a beküldések eredményét.

Hosszabb távon megfigyelhető a rendszer használatával, hogy az egyes feladatokat az évek folyamán milyen hatékonysággal oldották meg a tanulók, melyek voltak azok a tesztesetek, melyeknél a legtöbben hibáztak. Ezek felhasználásával a felkészítés még hatékonyabbá tehető.

A rendszer használata azért is előnyös, mert így a tanár az órán csak az algoritmusok tanítására koncentrálhat, az értékelés, a javítás automatikus. A hibás kódokat a diák saját maga is ki tud javítani a letölthető tesztadatok segítségével. Teljesen rossz megoldás esetén a letölthető megoldási javaslat a szöveges magyarázattal, pszeudó kóddal segíti a továbbhaladást.

A tesztesetek leírása, megadása további segítséget ad a tanuló számára, ezek rendszeres tanulmányozása a később feladatok megoldásában játszik fontos szerepet a tanuló fejlődésében.

A rendszer szándékosan nem tartalmaz feladatokat, mivel minden diák más ütemben tanul, máshol tart, esetleg más kategóriában versenyez. Nem szerencsés, ha olyan feladatok megoldásával próbálkoznak, amelyekhez még nem állnak rendelkezésre a megfelelő ismeretek. A diákok személyesen kapnak számukra szóló feladatot. Aki gyorsabban dolgozik, több feladatot tud megoldani, gyorsabb lesz a fejlődése, nagyobb sikereket érhet el. A rendszer használatával a folyamat még jobban felgyorsítható, automatizálható. Ezek a diákok a döntőkben már rutinosan használják az ottani programokat, például a biro.elte.hu-t.

Összefoglalás

A cikkben bemutattuk, hogyan lehet középiskolás diákok felkészítését segíteni különböző IKT eszközökkel, saját készítésű programokkal az országos tanulmányi versenyekre mind az alkalmazói, mind a programozói kategóriában. Ezekre azért van szükség, mert a tanítási órák számának csökkentése nem teszi lehetővé az órai felkészítést az iskolában, a tanári magyarázatokat, a megértést segítő példákat a diáknak otthon kell elolvasnia, megértenie. Az alkalmazói kategória esetében nincs túl sok lehetőség az oktatás, illetve a gyakorlás, értékelés automatizálására, IKT eszközökkel történő gyorsítására. Ellentétben a programozói versenyekkel, melyek esetében megmutattuk, hogyan lehet a programozási tanítását, a versenyekre való felkészülést automatizálni, gyorsabbá tenni, ezáltal a diákokat gyakorlottabbá tenni egyrészt saját, interaktív, kódolást tanító programmal (**kodolosuli.nejanet.hu)**, másrészt programkiértékelő rendszer (**progcheck.nejanet.hu**) használatával.

Ez utóbbi több ponton is eltér a versenyeken használt on-line értékelő rendszerektől. Az egyes tesztesetekre adott hibás válaszok esetén lehetőséget nyújt a hiba megkeresésére a tesztadatok letöltésével. Segíti a diákot saját programjának

ellenőrzésében, tesztelési terv elkészítésében, hiszen minden feladat esetén megmutatja, hogy a feladat készítői milyen tervet készítettek az ellenőrzésre. Nem megfelelő megoldás esetén a megoldás leírásával lehetőséget ad a tanulónak programjának újragondolására, átírására. Mindez automatikusan, önállóan történhet, felgyorsítva ezzel a gyakorlást, a fejlődést. A tanár továbbra is nyomon tudja követni a diák munkáját, fejlődését, látva a beküldés eredményeit akár segíthet is neki. A beküldések eredményét, a forráskódokat felhasználva még hatékonyabbá tudja tenni saját munkáját, melyből a következő nemzedék is profitálhat.

Irodalomjegyzék

[1] *Nemzeti Alaptanterv*, http://dokumentumtar.ofi.hu/index_NAT_informatika.html, 2014.
[2] Sándor Király, *How to teach computer programming if our goal is the International Olympiad in Informatics,* Teaching Mathematics and Computer Science, 9/1 (2011), 13–25.
[3] http://php.net/manual/en/index.php, 2014.
[4] http://nemes.inf.elte.hu/, 2014
[5] http://biro.inf.elte.hu/, 2014.
[6] *Ruby programming language*, https://www.ruby-lang.org/en/, 2014
[7] *Ruby on Rails web framework*, http://rubyonrails.org/ 2014