Online programozás a kódolósuliban

Balla Tamás, Király Sándor[2]

{[1]balla.tamas, [2]kiraly.sandor}@uni-eszterhazy.hu
Eszterházy Károly Egyetem
Matematikai és Informatikai Intézet

Absztrakt. A dinamikusan fejlődő IT iparág folyamatosan küzd a szakképzett szoftverfejlesztők hiányával, miközben a felsőoktatási intézményekben, az informatikai területen tanuló hallgatók száma alig növekszik, a lemorzsolódás viszont nagy. A létszám növelhető lehetne, ha a közoktatásban több diák kerülne kapcsolatba a programozással.

Az általunk fejlesztett és egy éve üzemeltett kodolosuli.hu portál célja, hogy ingyenes, interaktív kurzusokon keresztül bevezesse – elsősorban – a közoktatásban tanulókat a C++, C# és Java programozás alapjaiba. Az előadás célja az LMS rendszerben kialakított kurzusok módszertani áttekintése és a gyűjtött adatok elemzésével a portál hatékonyságának elemzése. A gyűjtött tapasztalatokra alapozva bemutatjuk a lehetséges, jövőbeni továbbfejlesztési irányokat. A megfogalmazott kutatási célok elsősorban arra irányulnak, hogy a tanulás minősége hogyan javítható az LMS rendszerbe épített intelligens algoritmusok (pl. neurális hálózat) segítségével.

Kulcsszavak: programozás oktatás, algoritmizálás, gamifikáció, interaktív tanulás

1. Bevezetés

Az informatikaoktatásban egyre fontosabb szerepet kap az algoritmizálási készség elsajátítása, valamint a problémamegoldó gondolkodás fejlesztése. Ha tekintjük a hatályos Nemzeti Alaptantervet, akkor tapasztalható, hogy ez a cél, ez a törekvés egyértelműen megfogalmazásra kerül. A Nemzeti Alaptanterv az alábbi módon határozza meg az informatika tudományterület közoktatásban követendő oktatási célját: *„Az informatikaoktatás célja a praktikus alkalmazói tudás, a készség és képességfejlesztés mellett a logikus, algoritmikus gondolkodás és a problémamegoldás tanítása."* (Magyar Közlöny 2012. évi 66. szám: Nemzeti Alaptanterv 10813 (181.o)). Az utóbbi cél elérésére az alapfokú és középfokú oktatási intézményekben a magasszintű programozási nyelvekhez kapcsolódó alapismeretek elsajátításával biztosítható leginkább, hiszen a tanulók motivációja növelhető azáltal, hogy a munkájuknak kézzel fogható eredménye van. Ugyanakkor nyilvánvaló az is, hogy a minimális számú (heti egyszer 45 perc) informatikaórák tematikájába nehezen, vagy egyáltalán nem építhető be a programozási alapismeretekkel kapcsolatos téma. Annál is égetőbbnek bizonyult ez a probléma, hiszen a munkaerőpiacról rengeteg programozó hiányzik, s nyilván nem lenne hátrány, ha már a középfokú oktatási intézmények orientálnák a tanulók a szoftverfejlesztői szakma irányába. Ehhez nyilván elengedhetetlen a tanulókat megismertetni a programozással a felsőfokú oktatási intézményekbe történő jelentkezés előtt. [1][2]

Ezt a problémát felismerve azt célt tűztük ki magunk számára, hogy készítsünk olyan kurzusokat, amelyekhez a tanulók ingyenesen hozzáférve online módon elsajátíthatják programozással kapcsolatos ismeretek a legnépszerűbb (C++, C# és Java) programozási nyelveken. Megvizsgáltuk a forgalomban lévő és ingyenes hozzáférhető LMS rendszereket abból a célból, hogy megállapítsuk, hogy melyiket alkalmazzuk a kurzusok készítése során, hiszen a fontos cél volt az is a kurzusfejlesztés során, hogy automatizált módon tudjuk ellenőrizni a kurzus teljesítése során

fejlesztett programokat, forrásrészeket. Azt tapasztaltuk, hogy nem áll rendelkezésre olyan LMS keretrendszer, amely mindhárom nyelven biztosítaná a gyakorlatorientált feladatok ellenőrzését. Ezért úgy döntöttünk, hogy saját keretrendszer építünk, amely biztosítja a fenti lehetőséget, valamint mindenképpen előnyös a későbbi továbbfejlesztések könnyebb implementálása miatt egy saját szoftverrendszer megvalósítása.

Ennek eredményeként elkészítettük a kodolosuli.hu portált, amely ingyenes formában (előzetes regisztrációt követően) online kurzusokat biztosít a tanulni vágyók számára C#, C++ és Java programozási nyelveken. A portált közel egy éve üzemeltetjük, cikkünk célja pedig a tapasztalatok bemutatása, valamint a jövőbeli fejlesztési irányok megfogalmazása. Célunk volt a portál, valamint a kurzusok kialakítása során, hogy a kurzusokban nagy hangsúlyt fektessünk a játékos alkalmazások bevonására, amelyek növelik a motivációt és segítik a gyorsabb és könnyebb feldolgozást, megértést, ezáltal hatékonyabbá teszik a tanítási-tanulási folyamatot. [3]

2. Kurzusok szerkezete

A kodolosuli.hu portál C++, C# és Java programozási nyelven kínál ingyenes, magyar nyelvű online kurzusokat. A keretrendszer megvalósítása során az LMS rendszert úgy készítettük fel, hogy a kurzusok fejezetek sorozataként reprezentálható, amely fejezetekben elhelyezett lecke vagy leckék tartalmazzák a tanítási-tanulási folyamatban átadni kívánt tényleges tudáselemeket. A leckék – típustól függően – tartalmazhatnak feladatokat, valamint játékprogramokat is, amelyek célja az elsajátított tudás gyakorlása, elmélyítése.

A fejezet célja, hogy a logikailag összetartozó tudáselemeket, leckéket egy egységbe foglalja, ahogyan az *1. ábrán* is látható. A három programozási nyelvhez készített kurzusokat úgy építettük fel, hogy a kurzus teljesítésével a tanulók alaposan felkészüljenek egészen az alaptípusoktól az eljárásorientált programozásig. A legfontosabb fejezetek a tananyagokban: az alaptípusok, változók, I/O műveletek, operátorok, elágazások, ciklusok, összetett adatszerkezetek (tömb, lista, struktúra, osztály), szövegfájlok kezelése.

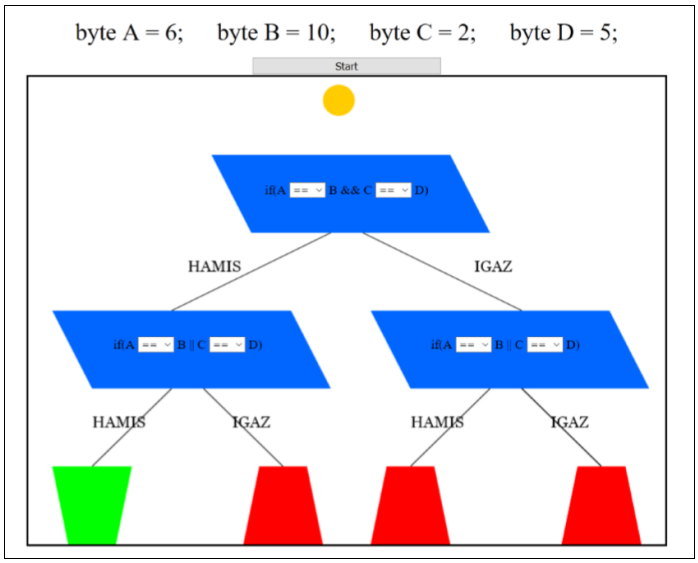
Programozási nyelvek ⇨ C#

Fejezetek listája

Sorszám	Cím	Fejezet leírása	Alfejezetek száma	Alfejezetek listája	Fejezet feldolgozása
1	Változók	Változók használata.	10	Alfejezetek	Feldolgozás
2	I/O műveletek	Hogyan lehet adatok beolvasni konzolról? És hogyan lehet a képernyőre írni adatokat?	6	Alfejezetek	Feldolgozás
3	Operátorok	A programozási nyelv által biztosított operátorok és használati eseteik.	5	Alfejezetek	Feldolgozás
4	Válaszút elé érkeztünk!	Eddig azt tapasztaltuk, hogy a program utasításai **egymás után kerülnek** végrehajtásra, azaz a végrehajtás, futtatás **szekvenciális**. Hogyan lehet azt megoldani, hogy például egy változó értékétől tegyük függővé utasítás vagy utasítások végrehajtását? Erről szól a fejezet.	3	Alfejezetek	Feldolgozás
5	Ciklusok	Az előző lecke után világos, hogyan lehet bizonyos utasításokat végrehajtani vagy nem végrehajtani egy feltételtől függően. A kérdés most az, hogyan lehetne bizonyos utasításokat **többször is végrehajtani** egy adott feltételtől függően.	10	Alfejezetek	Feldolgozás

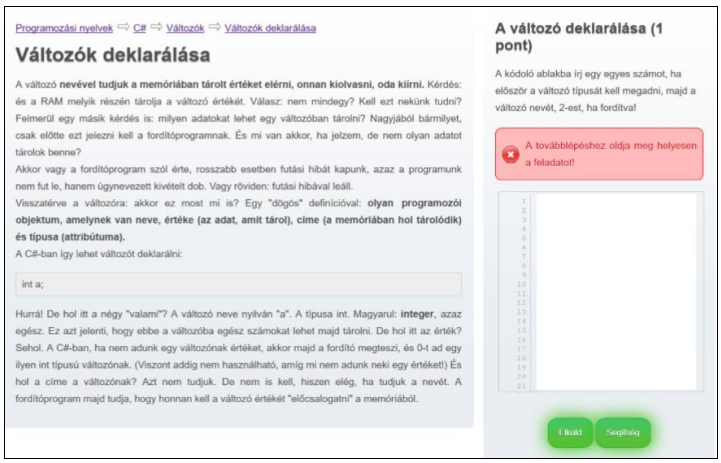
1. ábra: Fejezetek listája (C#)

A fejezetek a leckék lineáris sorozatával azonosíthatók. Három lecketípust különítünk el: tananyag feladatokkal, önálló feladat, önálló tananyag. Az önálló tananyagok tartalmazhatnak szövegblokkokat, a tanítási-tanulási folyamatot segítő audiovizuális elemeket, s itt kapnak helyet a játékprogramok is. A kutatások azt bizonyítják, hogy a gamifikált elemek a portálon növelik a diákok elkötelezettségét. [4] A tananyagokhoz több, az adott témakörre koncentráló gamifikált alkalmazás lett kifejlesztve, amelyek játékos feladatok keresztül motiválják a tanulókat, valamint elősegítik a megszerzett tudás diagnosztikus mérését, értékelését, s biztosítják annak lehetőségét, hogy a tanuló is meg tudja állapítani pillanatnyi tudásszintjét. A kifejlesztett alkalmazások nemcsak a motivációt növelik, hanem elősegítik a tananyag gyorsabb és hatékonyabb feldolgozását, megértést. [5] [6]



2. ábra: Az összehasonlító operátorok használatának gyakorlása

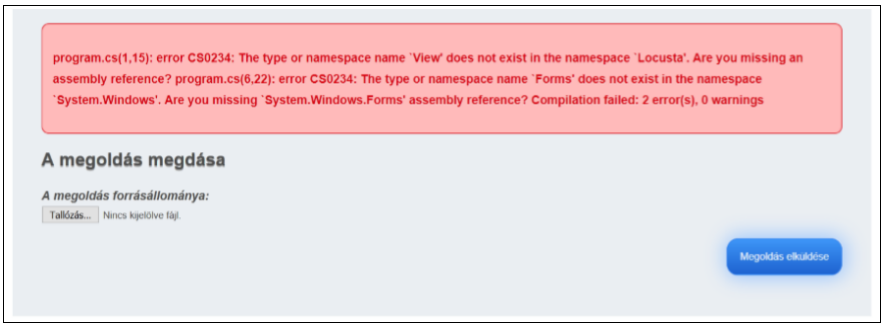
Lehetséges tananyagokat készíteni rövid feladatokkal, ebben az esetben a tananyag mellett, jobb oldalon jelennek meg a konkrét leckerész elsajátítását segítő rövid feladatsorok.



3. ábra: Egy lecke feladattal

A gyakorló feladatok kérdésbankból kerülnek kiválasztásra, így biztosított annak lehetősége is, hogy amennyiben a tanuló többször átismétli az anyagot, mindig új feladatot vagy feladatokat kapjon. A megoldandó feladatok száma függ az egyén választásától, valamint a kurzuskészítés során a szerkesztő tanár döntésétől, így a kurzusszerkesztő minimalizálja a megoldandó feladatok számát, de az adaptív környezetben a tanulóknak lehetőségünk van addig gyakorolni, míg nem rendelkeznek biztos tudással, gyakorlattal az adott témakörben.

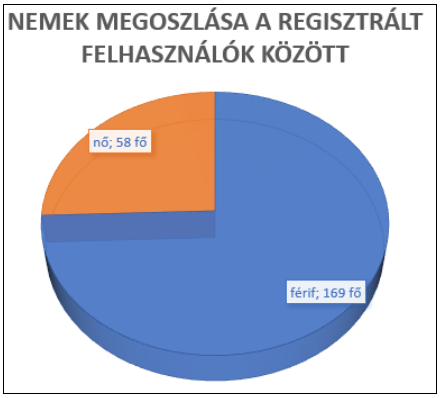
A harmadik típusú feladat az önálló feladat, amely esetén egy-egy témakör lezárásaként egy-egy komplex gyakorlati feladat megoldását várjuk el. A feladat megoldását a tanuló fejlesztőkörnyezet segítségével végzi, s a fejlesztés végeredményének forrását tölti fel a szerverre, amely lefordítja azt, s a program által meghatározott inputra adott output válaszok alapján eldönti, hogy helyes-e a megoldás. A programokat a rendszer futtatja, így fontosnak tartottuk, hogy bármilyen egyszerűek is a feladatok, a programok sandbox-ban fussanak hasonlóan a progcheck.nejanet.hu portálon használttal. [7] Ennek megfelelően 1 másodperces futási időlimitet állítunk be minden programnak, a memóriához és a merevlemezhez a programok nem férhetnek hozzá közvetlenül, az ezt próbáló programokat nem futtatja le a rendszer.



4. ábra: Az önálló feladat megoldásának egy lehetséges kimenete

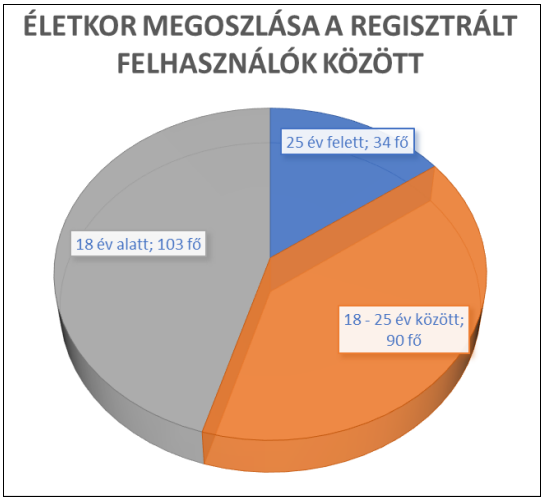
3. A portál használatának jellemzői

A kodolosuli.hu portálra több mint 200 tanuló regisztrált. A statisztikai mutatók, valamint a programozási magatartás elemzése szempontjából a felhasználóktól megköveteljük nemük, valamint születési évük megadását a regisztráció során. Ahogyan a következő ábra is mutatja, a portált nagyjából háromszor annyi férfi nemű felhasználó használja, mint nő.



5. ábra: Nemek szerinti megoszlás

A regisztrált felhasználók életkor szerinti megoszlását a következő diagram mutatja be: ahogyan látható is a felhasználók közel felét az alapfokú és középfokú közoktatási intézményben tanuló diákok teszik ki, emellett a portál szolgáltatásait használják egyetemisták is, valamint 34 fő 25 év feletti regisztrált felhasználó is aktívan használja a portál tananyagait, kurzusait.



6. ábra: Életkor szerinti megoszlás

A rendszer tárolja a teljes felhasználói aktivitást, azaz könnyen megállapítható, hogy ki, melyik feladattal, mennyi időt foglalkozott, s hányszor, hány hibával oldotta azt meg. Ezeket az adatokat áttekintve, a hibaforrásokat megállapítva, s ezekkel statisztikai számításokat végezve az alábbi fontos hibák állapíthatók meg:

- *A tanulók értő olvasásával problémák vannak.* Erre onnan következtetünk, hogy már a kezdetekben sem úgy oldják meg a felhasználók a feladatokat, mint ahogy az a feladat szövegében specifikálva van. A programozási stílus kialakítását célzó feladatoknál rendszerint problémát okoz, hogy a tanulók nem olyan szerkezetben írják be az utasításokat, mint ahogy a feladat azt elvárja.
- *Az összetett adatszerkezetek (listák, tömbök) megértése és kezelése nehézséget okoz.* A kurzust tanuló diákok nehezen barátkoznak meg az indexek használatával, az első próbálkozások alkalmával nem igazán értik meg az egy változóban több adat eltárolásának miértjét és módszerét. Többször meg kell erősíteni bennük, hogy az indexen keresztül juthatnak el a konkrét értékekig.
- *Érték és referencia típusok (C++ esetében mutatók) kezelése nehézséget okoz.* Egy-egy feladat megoldása során a tanulók nem minden esetben tudnak az érték és referencia átadás között különbséget tenni. Sok tanulónak nehézséget okoz eldönteni, hogy egy konkrét feladatmegoldás során melyiket kell használni és miért. Nem látja a két módszer közti markáns különbségéket, s nem érték miben rejlik a memóriacímek használatának előnye. Különösen problémás, amikor a témakör és az összetett adatszerkezetekkel kapcsolatos ismeretek kombinálni kell.

A problémák kezelésére nyilván át kell dolgozni a kialakított kurzusokat, hogy még markánsabban megtanítsa az elméleti alapokat, valamint a kérdésbank feladatait is ki kell bővíteni a minél hatékonyabb gyakorlati elsajátítás céljából. Célunk a kurzusok rendszeres felülvizsgálata és szükség szerinti átdolgozása. A tanulók együttműködését segítjük a fórumon keresztül, illetve a felületen lehetőségünk van kurzusszerkesztőknek is mentorként segíteni a tanulók munkáját.

4. Továbbfejlesztési irányok, lehetőségek

A korábban bemutatott elemzésre építve, valamint a portál üzemeltetése során gyűjtött tapasztalatokra alapozva a keretrendszernek, valamint a kurzusoknak fejlesztéseken kell átesniük. A fejlesztések mind azt a célt szolgálják, hogy a kurzusok feldolgozása és az egyes leckeelemek elsajátítása még hatékonyabb legyen. Emellett másik fontos cél, hogy az ellenőrző alrendszert is olyan irányba dolgozzuk át, hogy a rendszer által gyűjtött adatokat részletesebb tudjuk elemezni, s ezáltal a programozástanulás módját még inkább meg tudjuk ismerni.

Fontos cél az új kurzusok megvalósítása. Célunk, hogy mindhárom nyelvből készítsünk olyan haladó kurzusokat, amelyek az eljárásorientált világból vezetnek át az objektumorientált programozás alapjaiba. Nyilván ezzel együtt jár az új tananyagok, játékok és feladatok kidolgozása és megszerkesztése.

Egy másik fontos cél az lehet, hogy megvizsgáljuk, hogy a megszerkesztett tananyagok (kurzusok), a gyakorlást és az elmélyítést segítő feladatokkal együtt milyen hatékonyságot biztosítanak. Fontosnak véljük, hogy a keretrendszerben olyan funkciókat valósítsunk meg, amelyek segítségével a tananyag hatékonysága, valamint a tesztek, feladatok megbízhatósága is mérhetővé válik. Ezáltal a keretrendszer alkalmassá válik arra is, hogy olyan teszteket építsünk fel benne, amelyek valóban a kurzussal kapcsolatos tudást mérik. Itt szükséges a tesztelmélet témaköréhez tartozó számítási funkciók lefejlesztése, amelyekkel biztosítható a feladatok megbízhatósága, valódisága és objektivitása.

Másrészről fontos lehet az eszköztár bővítése is. Jelen pillanatban a felhasználók által megvalósított programokat olyan szempontból vizsgáljuk (önálló feladatok esetén), hogy egy előre meghatározott bemenetre milyen kimeneti eredményt állít elő. Kezdőknek nyilván elegendő egy-egy feladat ilyen szintű tesztelése is, de hosszabb távon sokkal használhatóbb megoldást ad, ha a forráshoz felépítjük a nyelvi szabályok alapján a kifejezésfát és azt vetjük előzetes elemzés alá. Az az hosszú távon képesek vagyunk azt is ellenőrizni, hogy egy-egy programban megtalálhatóak-e az előírt alprogramok a megfelelő paraméterekkel, vagy épp az új kurzusok esetén megállapítható, hogy létezik-e a megfelelő objektumosztály a megfelelő mezőkkel és metódusokkal. Ez a lépés a hatékonyságelemzés szempontjából is fontos lehet, hiszen a végcél nem csak a helyes, hanem a helyes és hatékony források megvalósítása. Az algoritmusok, programok hatékonyságelemzését klasszikusan három területen végezhetjük el:

1. A felhasználói és szerverprogramok esetén egyaránt fontos tényező a *végrehajtási idő*. Ez viszonylag egyszerűen mérhető jelen pillanatban is, kizárólag a program fordítás utáni indítási idejének és leállási idejének különbségét kell figyelembe venni.
2. *Tárigény*, amelyben azt vizsgáljuk, hogy a program mekkora memóriaterületet használ működése során. Ezt a jelenlegi implementációban nehéz vizsgálni, a jövőben meg kell teremteni ennek mérési metodikáját is a keretrendszerben.
3. *Bonyolultság*i mutatók, amely a megvalósított program algoritmusához rendel valamilyen bonyolultsági mértéket. Nyilván ennek elemzéshez a fenti kifejezésfa építésre van szükség.

Végül nagyon fontosnak tartjuk azt is, hogy a rendszer adaptivitását növeljük. Jelen pillanatban a kurzusszerkesztő a minimálisan helyesen megoldandó feladatok számát írja elő a tanulók számára, a tanuló pedig saját döntése alapján megoldhat további feladatok is a kérdésbankból. Amennyiben implementálunk tanulóalgoritmust (neurális hálózat vagy következtető motort), amely képes „megtanulni” a tanuló magatartását, úgy a rendszer képes a hatékonyabb működésre: felismerve a feladatok fontos elemeit képes az algoritmus eldönteni, hogy a tanuló valószínűen megtanulta-e már az anyagot, s ettől kezdve személyre szabott lesz a megoldandó feladatok

száma. Sőt, kapcsolati hálókat leírva feladatok, feladatcsoportok között még a lépés iránya is meghatározható. Ezáltal az adaptivitás fokozható.

5. Konklúzió

A cikk célja, hogy bemutassa a kodolosuli.hu webalkalmazás kurzusainak felépítését, amely egy olyan online, interaktív weboldal, amelyen C++, C#, és Java programozási nyelveken megtanítja a regisztrált felhasználókat a programfejlesztés alapjaira. A portál interaktív módon tanítja meg a programozás alapjait: bevezeti a diákokat a választott programozási nyelv elméleti alapjaiba, valamint a programozási gyakorlatok során megtanítja a diákoknak az alapvető programozási tételek implementációját.

A cikk második részében megvizsgáltuk, hogy a rendszer üzemeltetése során gyűjtött adatok alapján melyek azok a témakörök, amelyek a tanulóknak a legnagyobb problémát okozták a feldolgozás során. Kitértünk arra is, hogy a gyűjtött tapasztalatokra alapozva milyen jövőbeni fejlesztési irányokat tervezünk, képzelünk el.

Irodalom

1. Márk Edina: *Több mint húszezer informatikus hiányzik*, Világgazdaság, (utoljára megtekintve: 2017. október 4.), http://www.vg.hu/vallalatok/tobb−mint−huszezer−informatikus− −hianyzik−472181
2. Stubnya Bence: *Nem tüntetnek érte, de a jövő múlik rajta*, Index, (utoljára megtekintve: 2017. október 4.), http://index.hu/gazdasag/2016/03/04/informatikushiany_munkaeropiac_oktatas_informatika/
3. Michael Sailer, Jan Hense Heinz Mandl and Markus Klevers: *Psychological Perspectives on Motivation through Gamification*, Interaction Design and Architecture(s) Journal - N.19, 2013, pp. 28-37.
4. P. Fotaris, T. Mastoras, R. Leinfellner, Y. Rosunally: *Who wants to be a Pythonista? Using Gamication to Teach Computer Programming*.7th International Conference on Education and New Learning Tech-no-logies, Pages: 2611-2619.
5. Adrián Domínguez, Joseba Saenz-de-Navarrete, Luis de-Marcos, Luis Fernández-Sanz, Carmen Pagés, José-Javier Martínez-Herráiz: *Gamifying learning experiences: Practical implications and outcomes*. Elsevier, Volume 63, April 2013, pp 380–392.
6. Firas Layth Khaleel, Noraidah Sahari, Tengku Siti Meriam, Amirah Ismail: *The study of gamification application architecture for programming language course*. ACM IMCOM 2015 - Proceedings. Association for Computing Machinery, Inc, 2015. a17.
7. Sándor Király, Szilveszter Székely: *How to use our own program evaluation system to streamline teaching computer programming*. Teaching Mathematics and Computer Science 13:(1) pp. 73-80. (2015)