# A programozási alapismeretek oktatását támogató eszköz továbbfejlesztett változatának bemutatása

Menyhárt László, Dr Pap Gáborné

laszlo.g.menyhart@gmail.com,
papne@inf.elte.hu
ELTE IK

**Absztrakt.** Az Egyetemi képzésben az informatika szakosok számára a programozás oktatásának a teljes programkészítési folyamatot fel kell ölelnie. Tapasztalataink azt mutatják, hogy a hallgatók többségét ebből a folyamatból egyedül az érdekli, hogy a feladat megfogalmazásából kiindulva egy jól-rosszul összeállított, de működő kódot elő tudjon állítani. Egy korábbi cikkünkben bemutattunk egy olyan próbálkozást, ami a fenn vázolt problémának a feloldásában kíván segítséget nyújtani. Készítettünk egy alkalmazást, amely segítségével végigkövethető a teljes programkészítési folyamat. Az alkalmazásról kapott visszajelzések azt mutatták, hogy ötletünket érdemes továbbfejleszteni. A fejlesztést elvégeztük, lehetővé tettük pl. újabb összetett adattípusok használatát, több alprogramot tartalmazó project készítését és az egész alkalmazást áthelyeztük egy, a hallgatóság számára népszerűbb webes felületre. Jelen cikkünkben erről a fejlesztésről és az új alkalmazás fogadtatásáról kívánunk beszámolni.

## 1 Bevezetés

Az Eötvös Loránd Tudományegyetemen a Programozási alapismeretek nevű tárgy a Programozó Informatikusok és Informatika tanárok képzésében a programozás oktatásának alapozó tárgya. Ebben a tárgyban a teljes programkészítési folyamatot végigkövetjük a feladat megfogalmazásától kezdve a specifikáción és algoritmuson át a kódolás, tesztelés, hibakeresés lépésein keresztül egy működő és jól dokumentált program előállításáig. [2], [3]

Ebből a folyamatból a hallgató gyakorlatilag egyetlen lépéssel foglalkozik „szívesen” és ez a kódolás. Részben érthető, mert megír egy kódot, próbálgatja, javítgatja és végül sikerélményhez jut, mert megkapja a várt eredményt, esetleg még „csinosítja” programját. Ez azonban az amatőr programozók, nem a profi programozók szintje. Ahhoz, hogy később munkahelyükön, egy csapat munkájában megállják a helyüket, rá kell vennünk Őket a módszeres programozásra. Ehhez nyújt segítséget az elkészített alkalmazásunk használata. Használatával rávesszük a hallgatót, hogy fordítson figyelmet a specifikáció és az algoritmus elkészítésére még a kódolás előtt.

## 2 A korábbi (ProgAlapCppVarazslo) továbbfejlesztésének iránya az új PFW (Programing Fundamentals Wizard) alkalmazásban

Az előző cikkünkben bemutattuk a ProgAlapCppVarazslo-nak elnevezett program-csomagunkat. [6] [7] Akkor az derült ki felmérésünkből, hogy a hallgatók szívesebben dolgoznának webes felületen és grafikus megjelenéssel. Továbbá az akkori megszorításokat és továbbfejlesztési lehetőségeket is beépítettük az új rendszerbe.

## 2.1 Megjelenés

A legszembetűnőbb változás a megjelenés. A mai hallgatóság túlzottan is a Web világában él, ezért hogy nagyobb kedvet teremtsünk a PFW használatához, áttértünk az Microsoft Excel alapú tervezésről a Web alapú tervezésre.

## 2.2 Feladatok felépítésének átgondolása

Ahhoz, hogy ne csak egy nagyon egyszerű feladat megoldására legyen használható, lehetővé tettük a több modulból való építkezést. Jelen állapotában az alkalmazás a gépes zárthelyikben és a beadandó feladatokban előforduló számonkérési stílushoz készült. Ezek a feladatok mind olyan szerkezetűek, hogy van egy közös bemenet és több, a közös bemenettel és még plusz saját adatokkal dolgozó részfeladat, amiknek saját kimenetük van. Így a főprogram csak a közös adatok beolvasását és a részprogramok meghívását végzi. Minden részprogramnak külön meg kell adni a specifikációját és algoritmusát.

## 2.3 Hiányzó adattípusok pótlása

Kissé bonyolultabb feladatok megoldásához elengedhetetlen az összetett adattípusok alkalmazása, ezért lehetővé tettük a vektor és a rekord használatát is.

## 2.4 Nassi-Shneiderman Diagram (NSD)

Az oktatás menetét követve az algoritmus készítésében áttértünk a korábbi algoritmus-leíró nyelv használatáról a struktogram alapú tervezésre. [1] Több alkalmazással is lehet NSD-t rajzolni, melyeket Ben Shneiderman is összegyűjtött legutóbb 2003-ban. [10] Sajnos ezek közül már sok – több, mint a fele – nem érhető el. Pár nem elég rugalmas, van egy kettő, ami meg kódot is tud generálni. Nem említi viszont a luxemburgi Structorizert [11] és az idei évben egyik ELTE-s hallgató által fejlesztett StukiMania [12] weboldalakat.

# 3 PFW

Ez a vadonatúj webes alkalmazás próbál segítséget nyújtani a hallgatók számára, hogy könnyebben el tudják készíteni a beadandó vagy házi feladatukat és akár az összes probléma-megoldásuk vonalvezetőjeként szolgálhat. Az NSD szerkesztésen kívül az adatok összegyűjtését is támogatja. Az alkalmazás a [14] Internet címen érhető el. Azért, hogy a kommunikáció biztonságos HTTPS kapcsolaton keresztül történjen, tanúsítványra van szükségünk. A szerveren saját magunk által generált tanúsítvány viszont nincs aláírva fizetős hitelesítés szolgáltató által, így a böngésző figyelmeztetését tudomásul kell venni és tovább lehet lépni az oldalra. A keretrendszer filozófiája, hogy a megkötéseivel, mellyel rákényszerítjük a hallgatókat a gondolkodásra és, hogy az algoritmus készüljön el a kódolás előtt, azért segítsük is a munkájukat. Jó a hallgatóknak, mert segítséget nyújt az órákon kapott feladatok értelmezésében, hiszen amolyan vonalvezetőként szolgál. Rögzíteni lehet az ismert információkat az itt kialakított struktúra szerint. A tanároknak is jó lehet, mert egységes beadandók készülhetnek, azonos megjelenésű NSD-ket kell javítani, és a feladatok specifikációja illetve algoritmusa elektronikusan XML formátumban is bekérhető.

## 3.1 Bemutatás

Az alkalmazás angol és magyar nyelven érhető el. [14] A nyelvek közötti váltás a bal felső sarokban elhelyezett EN illetve HU linkekre történő kattintással lehetséges. Az alkalmazás ismertetője online is elérhető. [13]

### 3.1.1 Regisztráció és belépés

Az alkalmazás csak bejelentkezés után érhető el. Bejelentkezni háromféleképpen lehet: Gmail-vel OpenID segítségével, INF-es azonosítóval és az alkalmazásba történő regisztráció után.

Amennyiben a Gmail-t választja valaki, akkor engedélyezni kell az első bejelentkezéskor megjelenő Gmail-es oldalon, hogy az alkalmazás elérje a felhasználó email címét. Ezen kívül a felhasználó neve és a beállított nyelv információkat kapja meg a rendszer és automatikusan regisztrálja a felhasználót.

Ha valakinek van azonosítója az INF.elte.hu domainhez, akkor azt is használhatja. Ekkor ugyancsak automatikusan regisztrálja a rendszer a felhasználót a nevével és magyar nyelvvel.

Ha nincs, vagy nem szeretné használni valaki a fenti lehetőségeket, akkor előbb regisztrálni kell az oldalon, majd ezután be lehet lépni az új azonosítónak megadott email címmel és az oda kapott kezdeti jelszóval, melyet az első belépéskor meg kell változtatni.

Sikeres belépés után a központi oldalra jutunk, melyről ezután már nem is kell elnavigálnunk, hiszen ez egy egy-oldalas alkalmazás (Single-page application).

**Oldal felépítés és feladat struktúra**

A weboldal bal oldalán a menü található, jobb oldalán pedig a szerkesztő rész.

A menü dinamikusan változik aszerint, hogy a bejelentkezett felhasználónak van-e már elmentett feladata és, hogy megnyitotta-e már illetve az mennyi részfeladatot tartalmaz.

Amennyiben most lépett be először a felhasználó, akkor nincs elmentett feladata, így a következő menüpontok jelennek meg neki:

- Információ
- Új feladat
- Import
- Kilépés

Ha már van elmentett feladat a munkaterületén, akkor elérhetővé válnak a következő menüpontok is:

- Export
- Megnyitás
- Törlés

Új feladat létrehozása vagy megnyitás után átalakul a menü szerkezete így:

- Információ
- Mentés
- Mentés új verzióként
- Bezárás
- Általános
- Specifikáció
  - Bement
  - Előfeltétel
- Részfeladat 1.
  - Specifikáció
    - Bement

        - Előfeltétel
        - Kimenet
        - Utófeltétel
    - Algoritmus
- Tesztelés
- Fejlesztési lehetőségek

A Bezárás és Általános menüpontok között olvasható a feladat hosszú és rövid neve jelezve, hogy innentől nem alkalmazásszintű, hanem a feladattal kapcsolatos műveletek végezhetőek el.

A feladat több részfeladatot is tartalmazhat. Ilyenkor a Részfeladat 1. után még további menüpontok („Részfeladat 2.”, …) is megjelennek az almenükkel együtt.

**Menüpontok és műveletek ismertetése**

| Menüpontok kezdetben | Leírás |
|---|---|
| Információ | A menüpontra kattintva megjelenik egy rövid üdvözlő és leírást tartalmazó oldal. |
| Új feladat | Ezzel a menüponttal egy új feladatot lehet létrehozni, mely rögtön be is töltődik. |
| Import | Itt van lehetőség a weboldalról letöltött példát, vagy mástól/máshonnan kapott feladatot importálni a rendszerbe. |
| Export | A munkaterületünkön található állományok egy másolatát le lehet tölteni a szerverről. |
| Megnyitás | A felhasználó kiválaszthatja, hogy a hozzá elmentett feladatok közül melyikkel szeretne dolgozni és ez betöltésre kerül. Ekkor a menü pontok dinamikusan megváltoznak a nyitott feladattal végezhető műveletekre. |
| Törlés | Az elmentett feladatokat törölhetjük a szerverről. |
| Kilépés | Az alkalmazásból való kilépéskor kell ezt a menüpontot választani. |

| Menüpontok a feladat megnyitása után | Leírás |
|---|---|
| Mentés | A feladatok módosítása a memóriában (session-ben, az adott munkafolyamatban) történik. Adatbázisba csak e menüpont kiválasztásakor kerül be. A feladat egyedi azonosítója nem változik. |
| Mentés új verzióként | A feladatot lehet új verzióként is menteni, ekkor automatikusan legenerálásra kerül egy új név. |
| Bezárás | Ha már nem dolgozunk egy feladattal, akkor azt a munkát be kell zárni. |

| Menüpontok a feladat megnyitása után | Leírás |
|---|---|
| Általános | Ezen az oldalon van lehetőség a feladathoz tartozó alap információk beállítására. Egyedül a rövid azonosító megadása kötelező, mert ebből generálódik a feladat egyedi neve és szerepe van a generáláskor is a fájlok elnevezésekor. Ezen kívül a hosszú név, a feladat szövege is itt rögzíthető. Tárolhatjuk a környezetről ismert információkat és a szerző adatait is. |
| Specifikáció | A Specifikáció menüpontot kiválasztva automatikusan átkerül a vezérlés az első almenüpontra a Bemenet-re. Még egy másik almenü is van: előfeltétel. |
| Bemenet és Előfeltétel | Mivel olyan feladatok megoldására készült a PFW alkalmazás, amelyekben van egy közös bemenet és ezekre vonatkozó több részfeladat, ide a feladat szövegéből kigyűjtött közös bemeneti adatokat és a hozzájuk tartozó típusokat, kezdőértékeket, megváltoztathatatlan (konstans) tömb esetén a méretét, rekord esetén pedig a mezőket kell megadni. Az előfeltétel szintén a közös bemeneti adatokra vonatkozik, ennek megadása a jelen verzióban egyszerű alfanumerikus karakterekkel adható meg. Rövid mondattal vagy képlettel, mint az Excelben használt függvényeknél. |
| Részfeladat | Több részfeladat is lehet egy feladaton belül. Az ehhez tartozó alap információkat lehet itt megadni. Például a részfeladat jelét és szövegét. |
| Specifikáció (a részfeladaton belül) | A Specifikáció menüpontot kiválasztva automatikusan átkerül a vezérlés az első almenüpontra a Bemenet-re. Még három másik almenü is van: kimenet, előfeltétel és utófeltétel. |
| Bemenet és Kimenet | Mind a bemenethez, mind a kimenethez csak az adott részfeladat szövegéből kigyűjtött adatokat és a hozzá tartozó típusokat, kezdőértéket, megváltoztathatatlan (konstans) tömb esetén a méretét, rekord esetén pedig a mezőket kell itt megadnunk. |
| Előfeltétel és Utófeltétel | Az elő- és utófeltétel megadása is csak az adott részfeladatra vonatkozik. |
| Algoritmus | Ezen a lapon lehet az algoritmust megrajzolni kis dobozokból, az egyes elemeken történő jobb egér gomb hatására felbukkanó popup menü pontjainak segítségével. Az itt lévő pontok rövid leírását lásd később. |
| Tesztelés | Eltárolhatjuk a tesztelés információit. A bemenő adatokat, a tesztfájl nevét (amennyiben létezik), a várt kimenetet, a tényleges kimenetet és hogy sikeres volt-e a teszt. Ez utóbbi helyre szöveges mezőben rövid megjegyzést is lehet írni. |
| Fejlesztési lehetőségek | Itt lehet felsorolni a feladaton elvégezhető fejlesztési lehetőségeket. |

| Műveletek | Leírás |
|---|---|
| Szerkesztés | Az adott elem értékének szerkesztését indítja, vagyis az értéket bemásolja egy megfelelő szerkesztőmezőbe. Ez lehet egy egyszerű szöveges mező vagy hosszabb szöveges elem is. |
| Súgó | Ez egy kép, melyre az egeret ráhúzva az adott mezőről olvashatunk rövid leírást egy tooltip-ben. |
| OK | Szerkesztéskor az új érték elmentése a szerver oldalon a munkafolyamatban és az oldal frissítése történik. |
| Mégsem | Szerkesztéskor az új adatok elvetése és visszatérés a régi értékekhez. |
| Törlés | Az adott elem törlése. |
| Hozzáadás | Új elem hozzáadása alapértelmezett értékekkel. |
| Új részfeladat (link) | Új részfeladatot ad a feladathoz, majd újragenerálja a menüpontokat. |
| Részfeladat törlése (link) | Az adott részfeladatot törli. A menü pontjait dinamikusan újragenerálja. |
| NSD letöltése képként (gomb) | A megszerkesztett NSD képét PNG formátumban letölti a kliens gépre. |
| NSD szerkesztése (popup menü) | Amikor az NSD egyik dobozán az egér jobb gombjával kattintunk, akkor megjelenik egy popup menü a következő menüpontokkal. |
| Fel vagy le / | A menüpont kiválasztása azt eredményezi, hogy a doboz előző (vagy következő) doboz elé (vagy mögé) ugrik. |
| Beszúrás elé vagy mögé / | Ezt a menüpontot kiválasztva még egy almenü megjelenik, ahol a beszúrandó mintát kell kiválasztani. Ez lehet egy utasítás, elágazás, elöl tesztelő vagy számlálós ciklus illetve hátul tesztelő ciklus. |
| Szerkesztés | A doboz tartalmát egy felugró szerkesztő ablakban módosíthatjuk és menthetjük. |
| Törlés | A kijelölt doboz törlésre kerül. |
| Megjelölés | A kijelölt dobozt a rendszer megjegyzi, hogy azzal a következő két műveletet elvégezhesse. |
| Mozgatás | Az előzőleg megjelölt dobozt a most kijelölt elé mozgatja. |
| Másolás | Az előzőleg megjelölt dobozt a most kijelölt elé másolja. |
| Kód generálása (gomb) | A gombra kattintva letöltésre kerül a szerveren legenerált és ismert információkkal kitöltött Code::Blocks project a szükséges forráskódokkal egybe tömörített első verziója. [4] Ennek a fejlesztését kell folytatni a beolvasással, kiírással, stb. |

| Műveletek | Leírás |
|---|---|
| Dokumentáció generálása (gomb) | A gombra kattintva letöltésre kerül a szerveren legenerált és ismert adatokkal kitöltött dokumentáció első verziója. Ennek a szerkesztését kell folytatni a tartalomjegyzékkel, képernyő képekkel, stb. |

## 3.2 Technikai megvalósítás

Az alkalmazás Java nyelven lett implementálva és Glassfish alkalmazás szerveren fut. Az adatokat PostgreSQL adatbázisban tárolja. Felhasznált technológiák az OpenID, LDAP és email küldés. A kliens egy egyoldalas alkalmazás, mely AJAX-val kommunikál a szerverrel.

A feladatok az adatbázisban XML nyelven leírt formában vannak tárolva, mely betöltődik a memóriában és XPath illetve XSL transzformáció segítségével kerül megjelenítésre a különböző esetekben.

A feladatok leírásához saját definiálású kiterjesztett nyelvet használunk. TML-nek neveztük el, mint a feladat leíró nyelv (Task Markup Language). Ennek alapja az AML, melyet egy korábbi cikkemben mutattam be. [5]

# 4 Kérdések, válaszok

Miközben végiggondoltuk, elkészítettük és továbbfejlesztettük ezt a fajta segítségnyújtást a hallgatóknak, bennünk is felmerültek újabb kérdések, melyek közül néhányat itt meg is válaszolunk.

## 4.1 Az oktatás melyik fázisában mutassuk be a PFW alkalmazást a hallgatóknak?

Az első órákon, amikor még csak alapdolgokkal foglalkozunk, nincs értelme foglalkozni vele. Amikor viszont már az órákon is specifikálunk, algoritmust készítünk és ez alapján kódolunk, érdemes bemutatni. Előadáson ilyenkor már szó esik a tesztelésről és dokumentáció készítéséről is, ezeket élesben is kipróbálhatják a PFW használatával. Egyszeri bemutatása fontos, hiszen eleinte pl. nem ismerik a részprogram fogalmát és használatát, az alkalmazásban viszont találkoznak ezzel. Nem kell ekkor még belemenni a pontos magyarázatába, anélkül is tudják használni. Később viszont amikor már tanulják a függvényeket, talán könnyebben megértik a paraméterátadás fontosságát, hiszen ha jól készítették el egy feladat specifikációját, akkor a PFW azt már automatikusan megoldja.

## 4.2 A PFW kódgenerálása nem eredményezi-e azt, hogy a hallgatók nem tanulnak meg kódolni?

Ez a kérdés már a régi alkalmazásnál is felmerült, most még inkább azt válaszolnánk erre, hogy nem. Ennek több oka is van. A PFW célja sokkal több az automatikus kódgenerálásnál, inkább vonalvezetőnek és kódolási, dokumentálási segítségnek készül. Ezért elsősorban kezdő programozóknak ajánljuk, de nem csak az Ő munkájukat segítheti. [8] Az algoritmus készítésekor ragaszkodunk ahhoz az elvünkhöz, hogy az algoritmus nyelv-független és nem lehetnek benne nyelv-specifikus elemek. Ennek megfelelően a generált kódban nagyon jól fog majd tükröződni az algoritmus szerkezete, tehát az utasítások egymásutánja a szabályos C++ szintaxissal. Nem fognak viszont automatikusan megjelenni a C++ operátorok (pl := helyett =, és helyett && stb.), ezeket a hallgatónak kell a kódban kicserélnie. Nem lesz a generált kódban az ellenőrzött beolvasás és a kiírás, hiszen ezek nem részei az algoritmusnak. Ezek beépítése nélkül viszont nem működőképes a programjuk. Mivel az algoritmusunk nyelv-független, abban

a vektorok indexelése 1-től történik. Nagyon sokszor tapasztaljuk, hogy ez komoly fejtörést okoz a hallgatóknak a kódolásnál, hiszen a C++-ban át kell térni a 0-tól való indexelésre. Ez a feladat továbbra is rájuk vár. Végül még egy, a struktogramban használható alaptípusokból származó feladat hárul a hallgatókra. Nevezetesen: nincs külön struktogram elem a számlálós ciklusra, ezért az az algoritmusból elég érdekesen kerül át a generált kódba (While (i:=1..N)), amit természetesen a hallgatónak korrigálni kell. Vagyis, a PFW használata mellett is meg kell tanulniuk kódolni.

# 5 Alkalmazási tapasztalatok

A Hallgatóink egyelőre még csak ismerkedtek a felülettel, mert a kódolási részt tanulták eddig, október végén a kódolási ismereteket kértük számon a csoportokban egy zárthelyi dolgozattal. [13] [14] A következő időszak szól majd az algoritmizálásról, így nincs még túl sok visszajelzés. Ami van, az eddig pozitív vélemény volt.

Van egy olyan csoport, ahol angol nyelven folyik a több különböző országból (amerikai, brazil, török, vietnám, …) érkezett diák képzése. Ebben a csoportban különösen jó, hogy nem pszeudónyelven történik az algoritmus leírása, hiszen annak didaktikailag az anyanyelv lenne a pozitív hozadéka, míg az angol nyelv viszont nagyon hasonlít a programozási nyelvekre és nem tudjuk annyira kihangsúlyozni az algoritmus nyelvfüggetlenségét. [9]

Mielőtt elkészítettük volna ezt a cikket, szerettük volna munkánkat elbíráltatni felsőbb éves hallgatókkal is, akik már elvégezték ezt a tárgyat. Készítettünk egy kérdőívet azoknak, akik a tavalyi felmérésben már részt vettek, illetve azoknak, akik már elvégezték ezt a tárgyat, de még nem látták az előzőt. Ezeket a kérdőíveket egy-egy Google űrlapon válaszolhatták meg az érintett hallgatók. Nem vártuk el, hogy nevüket adják a válaszokhoz, bízván abban, hogy így objektív válaszokat adnak.

## 5.1 Felmérés – Első csoport

Ebbe a csoportba tartoznak azok, akik a múlt évben már részt vettek az akkori felmérésben és megadták az email címüket, hogy szívesen részt vennének a további kutatásban. Most ismertettük velük a fejlesztéseket és újra megkérdeztük a véleményüket. [15]

**Kérdések**

1. A szerver elérhető volt? (igen / nem)
2. Melyik időszakban próbálta ki az alkalmazást? (szöveges válaszát kérjük, pl.: 11 és 13 óra között)
3. Mennyi volt elégedett az alkalmazás válaszidejével? (gyors / megfelelő / elfogadható / lassú / nagyon lassú, leterhelt, kivárhatatlan)
4. használta volna ezt a segédeszközt, ha annak idején megkapja? (igen / nem)
5. Mennyire tartja jónak a webes és vizuális alkalmazás fejlesztésének irányát? (Csak így tovább! / Másik irány jobb lenne. Kérjük, írja le a fejlesztési javaslatoknál. / Nem kellene erőltetni.)
6. Talált-e hibát az alkalmazásban? Ha igen, akkor mit? (szöveges válaszát kérjük)
7. Milyen fejlesztési javaslatai lennének? (szöveges válaszát kérjük)
8. Egyéb észrevételek: (szöveges válaszát kérjük)

**Válaszok**

A múlt évben nagyon kevesen válaszoltak. 2-300 főnek lett kiküldve és összesen 13 választ kaptunk. Ebből 9 adta meg elérhetőségét, így csak őket tudtuk megkérdezni. Végül egy válasz érkezett, aki szerint a szerver 20 és 22 óra között elérhető volt, sőt gyors. használta volna az alkalmazást és további fejlesztést javasol. Sőt személyesen meg is keresett, hogy segít egy kicsit design-osabb külsőt varázsolni a rendszernek.

## 5.2 Felmérés – Második csoport

A második csoport kérdései nagyon hasonlóak voltak az előző évi kérdésekhez, csak egy kicsit fogalmaztuk át. [16]

**A Programozási Alapismeretek tantárgy beadandó feladatára vonatkozó kérdések:**

1. Mennyi időt töltött a beadandó elkészítésével? (órában)
2. Milyen százalékos megoszlás volt az algoritmus A= kódolás K= és dokumentáció D= között. Ha ez előzőek összege nem 100, akkor mire fordította a többi időt?
3. Milyen sorrendben készítette el az algoritmust (A), kódot (K) és dokumentációt (D)?

**A ProgAlapCppVarazslo kipróbálására vonatkozó kérdések**

4. Mennyire tartja jónak az ötletet? ( Nem látom értelmét / Jó az ötlet, de nagyon munkaigényes az alkalmazás / Jónak tartom / Nagyon jónak tartom)
5. Ön szerint segíti-e az algoritmuskészítés fontosságának erősödését? (Nem / Részben / Igen)
6. Ön szerint segíti-e a dokumentációkészítést? (Nem / Részben / Igen)
7. Lát-e valamilyen veszélyt abban, ha ezt az alkalmazást használják a hallgatók? (szöveges válaszát kérjük)
8. Zavarja-e, hogy az algoritmust kell előbb elkészíteni? (Igen / Nem)
9. Használta volna-e ezt a segédeszközt, ha annak idején megkapja? (Igen / Nem)
10. Hogy érzi, tudott volna időt megspórolni a beadandó elkészítésekor ezzel az eszközzel? (Nem készült volna el / Hátráltatott volna / Semennyit / Keveset / Sokat)
11. Talált-e hibát az alkalmazásban, és ha igen, akkor mit? (szöveges válaszát kérjük)
12. Milyen fejlesztési javaslatai lennének? (szöveges válaszát kérjük)
13. Egyéb észrevételek: (szöveges válaszát kérjük)

**A kérdőív kiértékelése**

A kérdőív linkjét az összes felsőbb éves hallgatónak kiküldtük, ami több száz, majdnem ezer hallgatót jelent. A tesztelés és a válaszadás mindössze 20-30 percet vett volna igénybe, körülbelül három hetes határidővel. Sajnos mindössze 2, azaz kettő hallgatótól kaptam választ.

1. Az egyik 6 órát foglalkozott összesen a beadandóval, míg a másik egy héten át napi 4-5 órát adott meg.
2. A százalékos megoszlás az algoritmus, kód és dokumentáció között úgy alakult, hogy az első hallgatónál 5-35-60% volt, míg a másodiknál 50-30-20%.
3. Mindketten betartották az Algoritmus-Kód-Dokumentáció sorrendet.
4. Az első nagyon jónak tartja, míg a másik hallgató szerint jó, de nagyon munkaigényes.
5. Szerintük egyértelműen illetve részben segíti az algoritmus kihangsúlyozását.
6. A dokumentációkészítést segíti, mindkettőjük szerint.

7. A második hallgató írt véleményt: „Mindössze annyit, hogy ha csak a Programozási alapismeretek tárgy keretében használnák a diákok az alkalmazást, és a későbbi tantárgyak esetében ezt a segítséget nem vehetnék igénybe, az talán nehézségeket okozna a kevésbé szorgalmas diákok számára. Mert ugye, a jót könnyű megszokni. :) De ha azokon az órákon/zh-kon, ahol lehet internetes segítséget használni, azokon ezt a programot is lehetne, akkor megérné a belé fektetett időt és energiát.”.
8. Nem volt zavaró a megkötés, miszerint az algoritmussal kell kezdeni.
9. Mindketten használták volna ezt a segédeszközt.
10. Az első sok időt, a második keveset tudott volna megspórolni.
11. Hibát nem jeleztek.
12. További fejlesztési lehetőséget nem írtak.
13. Egyéb véleményük sem volt.

Örülünk neki, hogy ez a pár válaszadó hasznosnak és az algoritmizálás fontosságát kiemelőnek találta munkánkat, bízunk benne, hogy nem a tárgy elvégzése után, hanem az oktatás aktuális pillanatában majd nagyobb érdeklődést kelt az új, webes PFW alkalmazás.

# 6 Fejlesztési lehetőségek

## 6.1 Elő- és utófeltételek

A specifikációban az elő- és utófeltétel a jelen verzióban egyszerű alfanumerikus karakterekkel adható meg. Rövid mondattal vagy képlettel, mint az Excelben használt függvényeknél. Ez így a generált dokumentációban nem szép, ezért ha igényes a hallgató, akkor a dokumentációban az egyenletszerkesztő segítségével kijavítja azt. Terveink között szerepel ezen probléma megoldására a MathML integrálása.

## 6.2 Tétel a tételben

Jelen pillanatban a PFW-ben nem megoldott a tétel a tételben típusú feladatok megvalósítása. A programozási alapismeretek tárgyban ezek még nagyon kis számban fordulnak elő, inkább a 2. félévben kapnak az ilyen típusú feladatok nagyobb hangsúlyt. Ugyanakkor tudjuk, hogy a felülről lefelé való programtervezés folyamatának oktatása ezt is tartalmazza, ezért tervezzük ennek megvalósítását is az alkalmazásunkban.

## 6.3 Beadás és kiértékelés

Lehetőséget biztosíthatunk az elektronikus formában való házi feladat beadásra PDF formátumban vagy email küldéssel a rendszerből. Akár egy elő-feldolgozás is lehetséges lenne, amivel kiszűrhetnénk a teljesen hibás vagy jó megoldásokat.

# 7 Irodalom

1. I. Nassi, B. Shneiderman: *Flowchart techniques for structured programming*, ACM SIGPLAN Notices, Volume 8 Issue 8, August 1973, Pages 12 - 26
2. Dijkstra E.W.: *A Discipline of Programming*, Prentice-Hall, Englewood Cli_s, 1973.
3. Szlávi Péter, Zsakó László: Módszeres programozás: Programozási bevezető, 18. Mikrológia
4. Sz. Csepregi, A. Dezső, T. Gregorics, S. Sike: *Automatic Implementation of Service Required by Components*, Workshop on Property Verification for Software Components and Services ,PROVECS 2007, http://lina.atlanstic.net/provecs/2007/provecs2007proceedings.pdf

5. Menyhárt László: *Can a language be before "the first programming language"?*, Teaching Mathematics and Computer Science, 2011, Volume IX, Issue II, 209-224 ISSN: 1589-7389, http://tmcs.math.klte.hu/Contents/2011-Vol-IX-Issue-II.html
6. Menyhárt László, Pap Gáborné: *Dokumentáció alapú programfejlesztés*; INFODIDACT 2012 konferencia, Zamárdi, Magyarország, 2012.11.15-2012.11.16.
7. *How can we get our students to think while we help their work too?: Document based development*; Proceedings of the 7th International Multi-Conference on Society, Cybernetics and Informatics. Konferencia helye, ideje: Orlando, Amerikai Egyesült Államok, 2013.07.09-2013.07.12. Florida: International Institute of Informatics and Systemics (IIIS), 2013. pp. 97-102. (ISBN:ISBN-13: 978-1-936338-83-2)
8. Vladimir Averbukh, Mikhail Bakhterev: *The analysis of visual parallel programming languages*, ACSIJ Advances in Computer Science: an International Journal, Vol.2, Issue3, No. 4,2013, ISSN : 2322-5157
9. Martin Weise: *A Model for Teaching Informatics to German Secondary School Students in English-language Bilingual Education*, Proceedings of the 6th International Conference ISSEP 2013; Oldenburg, Germany, February 26–March 2, 2013/Diethelm et al. (Eds.)/ Potsdam: Universitätsverlag Potsdam, 2013/ S.127-137
10. B. Shneiderman: *A short history of structured flowcharts (Nassi-Shneiderman Diagrams)*, University Maryland, 2003, http://www.cs.umd.edu/hcil/members/bshneiderman/nsd/
11. Bob Fisch: *Structorizer*, http://structorizer.fisch.lu/
12. Molnár Tamás: *StukiMania*, http://stukimania.hu, 2013
13. http://xml.inf.elte.hu/2013_14_1/progalap/anyagok/PFW_leiras.pdf
14. https://157.181.166.134:8181/PFW/index2_HU.jsp
15. https://docs.google.com/forms/d/1LfhO64yJRFhywHqRBfoJzawK8PBZDizImWkTFnI4HmA/viewform
16. https://docs.google.com/spreadsheet/viewform?usp=drive_web&formkey=dERsTi1rS1V6OGc3Rzc3cm85YW5KNFE6MA#gid=0