VIZUÁLIS MÓDSZEREK OKTATÁSÁNAK HATÁSA A HALLGATÓK PROGRAMOZÁSI HIBÁIRA

Johanyák Zsolt Csaba[1], Tóth-György Ferenc[2]
[1,2]Főiskolai adjunktus
[1,2]Kecskeméti Főiskola, GAMF Kar, Kalmár Sándor Informatikai Intézet
[1]johanyak.csaba@gamf.kefo.hu, [2]tgyferi@hetenynet.hu

Absztrakt: Főiskolánkon a hallgatóink a C és C++ nyelven keresztül ismerkednek meg a programozással és szoftverfejlesztéssel. Ezt a választást a főiskolai profilunk indokolja. A kikerülő hallgatók főleg rendszer közeli programokkal találkoznak, és ezeket fejlesztenek. Az utóbbi időben sok bírálat ért bennünket, hogy ezek a nyelvek nem alkalmasak a kezdők oktatására, mivel rendkívül nehéz a szintaktikájuk és profiknak tervezték őket, ezen kívül programozást kell oktatnunk és nem nyelvet, míg a C és C++ oktatás úgy tűnik mások számára, hogy csupán a szintaktikáról szól, sokan a Java vagy C# nyelvet javasolják bevezetni a C/C++ helyett. Ezen kívül a képzésünk jelentős részében nem vizuális módon fejlesztünk, hanem parancssoros programokat íratunk sorról-sorra történő begépeléssel. Elsősorban saját kollégáink szerint a programozás már nem egyenlő a gépeléssel, a vizuális fejlesztés is programozás, sőt manapság egyre inkább erre és ezzel kellene felkészíteni a diákokat. Sok szempontból egyetértünk ezekkel az elképzelésekkel, és objektív módon megvizsgáltuk, hogy a vizuális és nem vizuális fejlesztés hogyan hat a hallgatóink képességeire.

BEVEZETÉS

Néhány éves programozás-oktatási, főleg gyakorlatvezetési tapasztalattal könnyen megfigyelhető, hogy bizonyos tervezési és implementálási hibák minden félévben felbukkannak a hallgatók órai vagy egyéni munkája során. A vizsgaidőszakban hétről hétre több száz programozási feladatot javítunk ki és értékelünk. Az általunk tapasztalt hibák többsége néhány olyan okra vezethető vissza, mint a gyakorlatlanság, a kontakt órán kívüli munka teljes hiánya vagy annak elégtelensége, türelmetlenség, a fejlesztőrendszer szolgáltatásainak hiányosságai, stb.
Tanulmányunkban egy a hallgatók körében végzett kérdőíves felmérésből kiindulva szeretnénk megvizsgálni azt, hogy a vizuális alkalmazásfejlesztési módszerek oktatása milyen változásokat hozott a hallgatók programozási hibáinak gyakorisága és eloszlása terén.
A felméréshez két kötelező tárgyat választottunk, amelyek legtöbbször időben egymást követve jelentkeznek a hallgatók tanrendjében. Mindkettő objektum orientált programozást tartalmaz. Az első a szabványos C++ konzolalkalmazásokról szól, a második a C# nyelvre és a fejlesztőrendszer által biztosított vizuális eszközök erőteljes alkalmazására épül.
Dolgozatunk első részében röviden áttekintjük szoftverfejlesztéssel kapcsolatos tantárgyakat a Kecskeméti Főiskola GAMF Karának mérnök-informatikus képzésében megvilágítva a vizsgálatban érintett két tantárgy elhelyezkedését a mintatantervben. Ezt követően áttekintjük a kérdőíven felsorolt jellegzetes hibák főbb típusait. A dolgozat harmadik szakaszában összehasonlítjuk és elemezzük a két hallgatói kör által adott válaszokat. Cikkünk utolsó részében megfogalmazunk néhány módosítási elképzelést, amelyek alkalmazásától javulást remélünk.

PROGRAMOZÁS OKTATÁS A KECSKEMÉTI FŐISKOLA GAMF KARÁN

Az oktatás során átadott programfejlesztéssel kapcsolatos tudásanyag és megszerezhető készségek szempontjából alapvetően két szintet különböztethetünk meg képzésünkben. Az alapozó szinthez három kötelező tárgyat sorolunk, melyeket a mintatanterv szerint haladó mérnök-informatikus hallgatók három félév alatt sajátíthatnak el. A haladó szintet három kötelező és három szabadon választható tárgy alkotja. A vizsgálatban érintett mindkét tantárgy (*Programozási paradigmák és technikák* és *Vizuális programozás*) ebbe a kategóriába tartozik.
Az alapozó szint első félévében hallgatóink a *Problémaosztályok, algoritmusok* című tantárgy keretében elmélyítik az algoritmus fogalmát, és megismerik a különböző algoritmus leíró módszereket. A második félévben *Programozás I.* címmel a kapnak bevezetést a C nyelv alapjaiba heti két óra előadás és két óra laborgyakorlat formájában. Ezután következik a *Programozás II.* című tantárgy heti két óra előadás és két óra laborgyakorlat formájában, ahol továbbra is a C nyelven illetve a C++ nem objektum orientált elemein keresztül a fájlszerkezetekkel, adatszerkezetekkel ismerkednek.
A haladó szintet a *Programozási paradigmák és technikák* című tantárgy nyitja meg, ahol a C++ nyelv segítségével az objektum orientált programozási (OOP) alapok elsajátítására kerül sor heti két óra előadás és két óra laborgyakorlat formájában. Az előzőekben említett négy kurzus sikeres teljesítése után javasoljuk a programozás szigorlat felvételét, bár ez utóbbi a programozási tárgyak követelményeinek sikeres teljesítése nélkül is megkísérelhető. A *Programozási paradigmák és technikák*at kötelező módon a *Vizuális programozás* követi, amelynek keretei között a hallgatók megismerkednek a vizuális technikákkal egy gyors alkalmazásfejlesztést támogató magas szintű fejlesztőeszköz (Visual Studio 2005 Professional) és egy újabb objektum orientált nyelv (C#) elsajátításán keresztül heti két óra előadás és két óra laborgyakorlat formájában. Ezzel párhuzamosan jelenik meg a tervezésre összpontosító *Szoftvertechnológia* című tantárgy heti két óra előadás formájában. A haladó szinthez kapcsolódó oktatás lezárásaként szabadon választható módon a hallgatók *ASP.NET*, *Java* vagy *PHP* alapú szerver oldali web-programozási ismereteket szerezhetnek a korábbiakhoz hasonlóan egy-féléves heti négy órás tárgyak keretében. Ezt kiegészítve a *Hálózati és Web technológiák* szakirány hallgatói kötelező jelleggel tanulnak *Java* alapú szoftverfejlesztést.
A tematika kialakítása során az alapozó szinten és részben a haladó szinten is igyekeztünk minél általánosabb, nem elavuló tudásanyagot összeállítani és átadni hallgatóinknak, melyet akár egész szakmai pályafutásuk alatt sikerrel felhasználhatnak. Ezért esett a választás az alapozó képzésben a C és C++ programozási nyelvre, hiszen az alapszoftvereket és operációs rendszereket mind a mai napig főleg ebben fejlesztik, és úgy tűnik egyelőre nincs olyan programnyelv, mely képes lenne ezt a feladatot átvenni tőlük. A vizuális alkalmazásfejlesztés tekintetében a hallgatói igényeket követve a C# alapú .NET programozás mellett tettük le voksunkat. A szerver oldali web-programozással kapcsolatos kínálatunkkal igyekeztünk minél szélesebb körben lefedni a fontosabb irányzatokat.

A FELMÉRÉS ELŐKÉSZÍTÉSE

Felmérésünk előkészítése során két célt tűztünk ki magunk elé. Egyrészt szerettünk volna tisztább képet kapni arról, hogy melyek azok a hibák és problémák, amelyek gondot okoznak hallgatóinknak a programfejlesztés elsajátítása és hétköznapi programozási gyakorlatuk során. Másrészt szerettük volna választ kapni arra a kérdésre, hogy a közkedvelt vizuális technikák oktatása gyakorol-e hatást ezen hibák előfordulására, esetleg a kiküszöbölt problémák után maradó „űrt” újabb hibákkal tölti-e fel.

A felmérés legegyszerűbb módszereként kérdőíveket használtunk. A kérdéssor első változatát saját tapasztalatokra és órai jegyzetekre támaszkodva készítettük el. Az „órai jegyzeteket” itt úgy kell értelmezni, hogy a gyakorlatok során feljegyeztük a jellegzetes hibákat, problémákat és félreértéseket. Az előkészítés következő szakaszába bevontuk a többi gyakorlatvezetőt valamint a hallgatókat is. Informális beszélgetések keretében gyűjtöttük a további relevánsnak tűnő kérdéseket. Az összegzést és rendszerezést követően egy negyven kérdésből álló kérdőív állt össze. A könnyű kezelhetőség/feldolgozhatóság és gyors kitölthetőség érdekében a kérdéseket úgy fogalmaztuk meg, hogy egy 1 és 10 közötti egész szám megadásával, pontosabban bekarikázásával, legyenek megválaszolhatóak. Általában az 1-es érték a „soha” válasznak, míg a 10-es érték a „mindig” válasznak felel meg.

A kérdéseket három csoportba szerveztük a vizsgálatban részt vevő hallgatói kör figyelembe vételével. Az első csoport 18 kérdése olyan problémákkal foglalkozik, amelyek mindkét nyelv és alkalmazásfejlesztési stílus elsajátítása és használata során előfordulnak. Az alábbi felsorolás két jellegzetes kérdést mutat be az első csoportból.

- Egy azonosítót egy helyen nagybetűs névvel, míg máshol kisbetűs névvel használok.
- Blokknyitáskor nem gépelem be rögtön a záró zárójelet is, és így egy idő után „elveszek” a zárójelek között.

A második csoportba soroltuk azt a 14 kérdést, amelyek csak a C/C++ nyelvpároshoz kapcsolódnak. Ebbe a csoportba tartozik például a következő két kérdés.

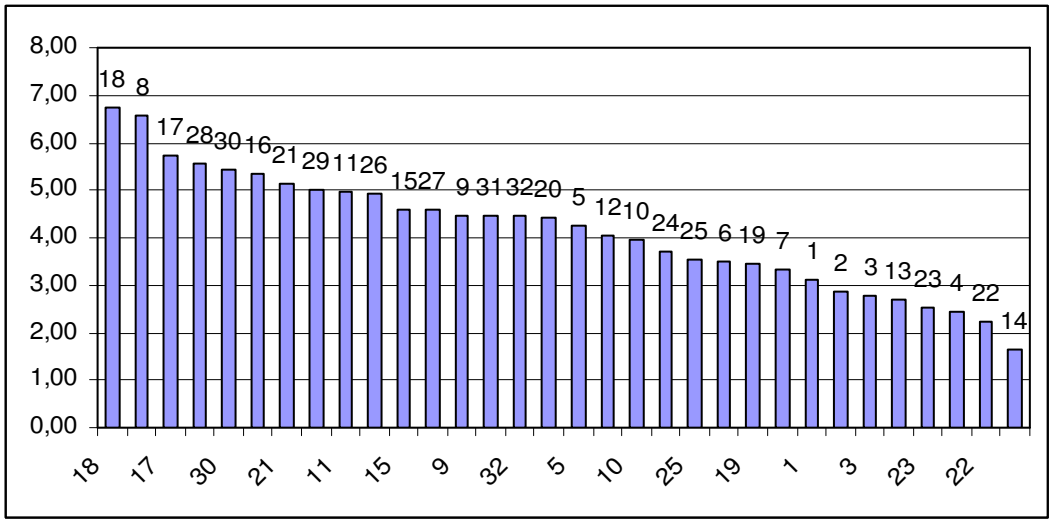
- Az = operátorral adok értéket karakterlánc típusú változónak.
- A lefoglalt memória felszabadítása után nem szoktam kinullázni a pointereket C/C++-ban.

A kérdések harmadik és egyben legkisebb csoportja 8 kérdéssel a vizuális alkalmazásfejlesztéshez és a C# nyelvhez kapcsolódik. Ebbe a csoportba tartozik például a következő két kérdés.

- Az eseménykezelő vázát nem automatikusan hozom létre, hanem begépelem a teljes eseménykezelő metódust anélkül, hogy feliratkoztatnám azt az eseményre.
- A feleslegessé vált vagy véletlenül létrehozott eseménykezelőt kódszerkesztőben törlöm ki, majd fordításkor hibát kapok, mivel nem töröltem a metódus regisztrációját az eseményre.

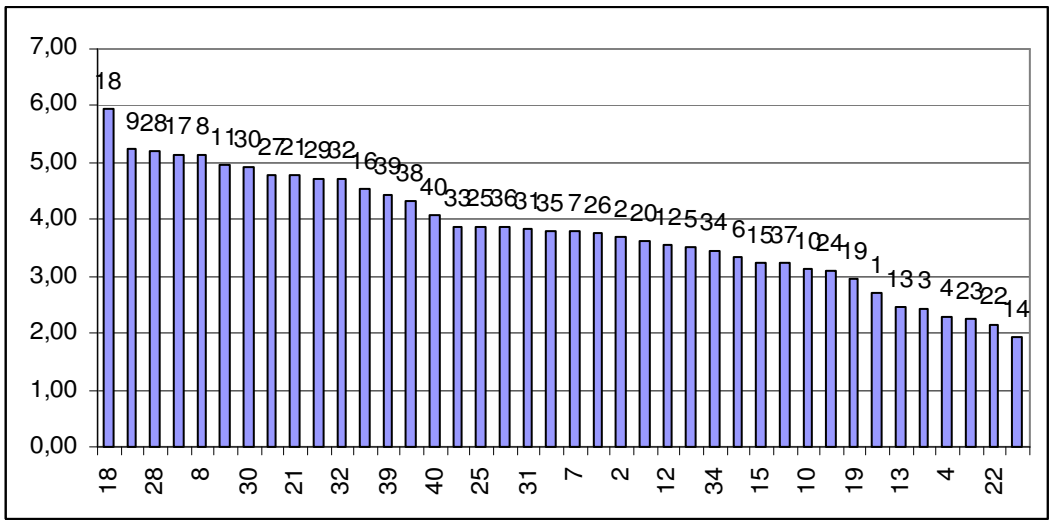
FELMÉRÉS ÉS KIÉRTÉKELÉS

A kérdőívek kitöltése önkéntes alapon történt. A felmérésben 172 hallgató vett részt, közülük 79-en a *Programozási paradigmák és technikák* tárgy hallgatói és 93-an (a tárgyat felvevő hallgatók 100%-a) ismerkedtek a *Vizuális programozás*sal. Ennek megfelelően az első hallgatói kör csak az első két kérdéscsoporttal foglalkozott. A második hallgatói csoport már mindkét tantárggyal közelebbi kapcsolatba került, így ők minden kérdésre válaszoltak. A válaszokat kiértékelő 1-3 ábrák vízszintes tengelyén az egyes kérdések sorszámai a függőleges tengelyen a hallgatók által adott értékelés szerepel.

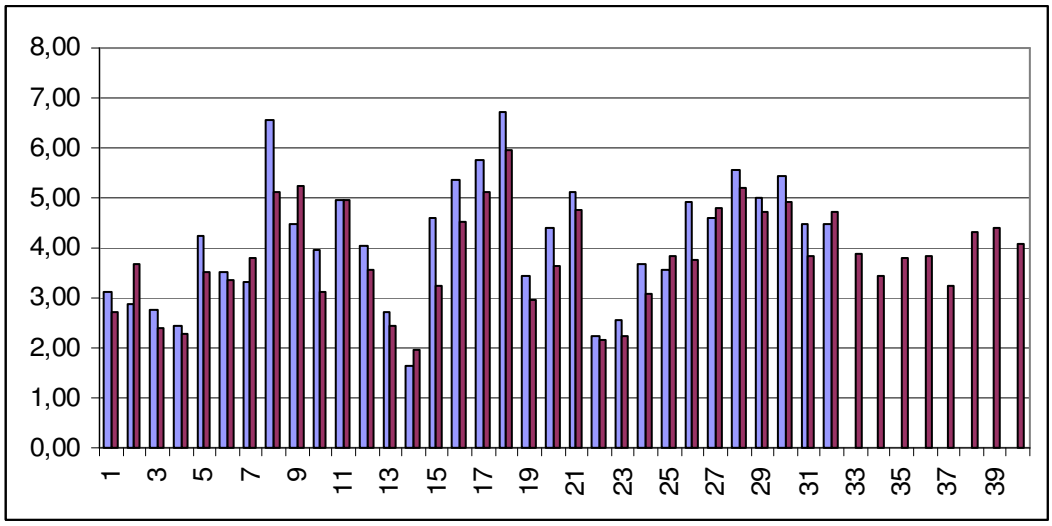


1. ábra
Csökkenő sorba rendezett hibák az első csoport esetén

A kérdőívek kiértékelése több tekintetben meglepő eredményeket hozott. Mindkét hallgatói csoport esetében a leggyakrabban előfordulónak ítélt probléma a függvénykönyvtárak fogalmának tisztázatlansága volt. Mivel a témakör részletesebb ismertetése előadáson történik, így összefüggés feltételezhető az előadások gyér látogatottsága és ezen eredmény között.
Az első csoportnál (ld. 1. ábra) második és harmadik helyen gyakorlati jellegű gondok jelennek meg. Ezek az automatikus kódkiegészítés és a nyomkövetés (lépésenkénti programvégrehajtás, változók értékének követése, stb.) használatának elmaradása. Véleményünk szerint ezek olyan készség szintű hiányosságok, amelyek visszavezethetők a kontakt órán kívüli gyakorlás elégtelen mértékére. Az automatikus kódkiegészítés alacsony szintű kihasználtságában valószínűleg az is szerepet játszik, hogy a hallgatók gépelés közben a billentyűzetet nézik és nem a képernyőt.

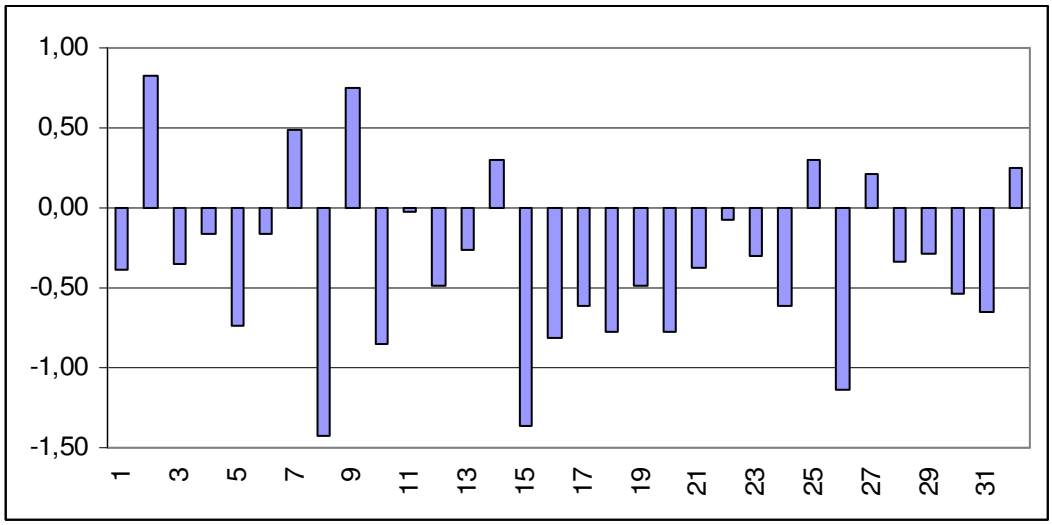


2. ábra
Csökkenő sorba rendezett hibák a második csoport esetén



3. ábra
Összehasonlító oszlopdiagram

Az első csoport következő három problémás területe szintén jellegzetesen elméleti jellegű kérdés. A dinamikusan lefoglalt területek felszabadítása, a fejléc-állományok szerepe és használata illetve a programmodulok kialakításának szükségessége. Ezen problémák „előkelő” helyezése jelzi, hogy a képzés gyakorlati részében is nagyobb hangsúlyt kell helyezzünk a tervezés bemutatására és a módszeres alkalmazásfejlesztés fontosságára.
A vizuális módszereket alkalmazó második csoportnál (ld. 2. ábra) a hibák gyakorisága kevés kivételtől eltekintvecsökkent (3. és 4. ábra), és a kritikus területeket jelölő élbolyban kis mértékű átrendeződés figyelhető meg. Hangsúlyozottabban jelennek meg olyan újabb témakörök, amelyek nagyrészt elméleti ismeretekhez köthetők. Így például kiemelkedik C# nyelvnél a referencia és érték típusok és C++-nál a mutató és érték típusok összekeverése.



4. ábra
Hibák előfordulási gyakoriságának változása

A hibák előfordulási gyakoriságának változását bemutató 4. ábrán jól megfigyelhető, hogy a problémák többségénél az előfordulás gyakoriságának csökkenését eredményezte a vizuális módszerek bevezetése. A vízszintes tengelyen megjelenő számok itt is az egyes kérdések sorszámát azonosítják. A 32 közös hibatípusból csupán hét esetben fordult elő, hogy a szubjektív megítélés alapján meghatározott előfordulási mérték növekedett. Ilyen gond volt, hogy egy azonosítót egy helyen kis betűvel, máshol nagy betűvel ír a hallgató. Ez arra vezethető vissza, hogy kellő gyakorlat hiányában kevéssé vált megszokottá osztálydiagramok használata.
Az automatikus kódkiegészítés használata javult a legnagyobb mértékben a vizuális technikáknak köszönhetően, azonban ennek is lehetnek hátrányos hatásai. Így például azon hibák előfordulása, ahol egy for ciklus fejlécében a ciklusváltozó definiálásának elmaradása esetén az eredetileg *i* nevet viselő ciklusváltozóból a fejlesztőrendszer int típusnevet generált.
A második legnagyobb javulási mértéket „Nem szoktam kezdőértékkel ellátni a változóimat" hibánál tapasztaltuk. Ez nem igazán a vizuális módszereknek köszönhető, hanem annak, hogy C++-tól eltérően a C# nyelvben a lokális változók kezdőérték adásának elmaradása fordítási hibát eredményez, így a futási idejű komolyabb probléma szintjére nem jut el a hiba.
További jelentős javulás figyelhető meg a fordítási hibák kezelése és értelmezése területén. Ez részben annak is betudható, hogy .NET alatt a fordítási hibaüzenetek magyar változata jelenik meg, amennyiben a nyelvi csomagot telepítettük.

KÖVETKEZTETÉSEK

A felmérésünk szerint a nem vizuális jellegű fejlesztés és programozás oktatás alkalmas a szintaktika elsajátítására, hiszen ezzel volt a legkevesebb problémája a hallgatóknak, annak ellenére, hogy a C/C++ ebből a szempontból igen nehéz. Azonban nem alkalmas az elmélet elmélyítésére heti két gyakorlati órát feltételezve, ugyanis 90 perc alatt nincs mód több modulból álló saját fejléc állományokkal működő szoftvereket fejleszteni úgy, hogy mindent kézi erővel gépelnek be a diákok. Ezen problémán csak az otthoni beadandó programok segíthetnének. Ugyan ez a helyzet az OOP lényegének a megértésével és elmélyítésével is. Túl kisméretű feladatokat lehet csak gépelési technikával megoldani, amiből nem látszódik az egész lényege, értelme.
Meglepő eredményt mutat, hogy nem tudnak a hallgatók szoftvert tesztelni, sőt még nyomkövetni sem. Ennek a technikának az oktatására külön oda kell figyelnünk a későbbiekben.
A vizuális fejlesztést hallgató diákjaink lényegesen többet tudnak a programok szerkezetéről, működéséről. Köszönhető ez a tervezés megjelenésének a tananyagban és annak, hogy a fejlesztő eszközök automatikusan modulokra tördelik a programokat, illetve UML diagramból legenerálják magát a program szerkezetet és viszont.
Sokak szerint a vizuális szoftverfejlesztéssel nem lehet programozás oktatást kezdeni, mert a fejlesztő eszköz elsajátítására megy el a drága és kevés idő, és közben nem tanulnak meg programozni. A mi tapasztalataink azt mutatják, hogy a programozás manapság valóban nem gépelést jelent, hanem komoly elméleti alapokra helyezett szerkezet tervezést és kódgenerálást. A hagyományos fejlesztéssel a szintaktikát meg lehet tanítani, de csak olyan kis méretű feladatokat lehet megoldani, amin keresztül az elmélet sem mélyül el, sőt értelmetlennek látszódik.