Západočeská univerzita v Plzni

FAKULTA PEDAGOGICKÁ

KATEDRA VÝPOČETNÍ A DIDAKTICKÉ TECHNIKY

PROGRAM PRO VALIDACI SEMESTRÁLNÍCH PRACÍ PŘEDMĚTU PROGRAMOVÁNÍ 1

BAKALÁŘSKÁ PRÁCE

Jiří Ulrich

Informatika se zaměřením na vzdělávání

léta studia (2007 - 2012)

Vedoucí práce: *Mgr. Tomáš Jakeš*

Plzeň, 29. červen 2012

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně s použitím uvedené literatury a zdrojů informací.

Plzeň, 29. červen 2012

……
vlastnoruční podpis

OBSAH

1 ÚVOD

Cílem této práce je popis a řešení problému opravy studentských zápočtových prací předmětu Programování 1 v bakalářském studijním programu Informatika se zaměřením na vzdělávání. Hlavním problémem opravy studentských prací je jejich vysoký počet a velká časová náročnost spojená s jejich evidencí a kontrolou.

Obsahem je podrobný rozbor a analýza současného stavu z pohledu vyučujícího. Je zde rozebrána struktura předmětu a podmínky pro úspěšné udělení zápočtu, které dále rozebírá kapitola věnována popisu struktury prací, které student v průběhu semestru odevzdává a jež jsou podmínkou pro úspěšné splnění předmětu.

Další důležitou kapitolou je popis současného postupu při opravě a evidenci prací. Zde je kladen důraz na přesné popsání všech dílčích úkonů, které vyučující provádí při přijetí nového příkladu k opravě. Následující kapitola se zabývá způsobem hodnocení chyb v jednotlivých sekcích, které jsou součástí studentské práce, a na ni přímo navazuje kapitola s kategorizací nejčastějších chyb, kde je popsáno několik základních chyb, kterých se student dopouští při zpracování samostatné práce.

Popsány jsou mimo jiné možnosti automatické validace a evidence prací, které by měly usnadnit a hlavně urychlit jejich opravu, aniž by došlo ke ztrátě individuálního přístupu ke studentům. Součástí kapitoly je také podkapitola zabývající se možnostmi odhalení podvodů.

Poslední kapitoly se zabývají popisem programu, který řeší několik základních problémů a činností, které je možno plně či alespoň částečně automatizovat. Důraz je kladen na jeho snadnou možnost editace a dodatečné rozšíření.

2 ANALÝZA A POPIS SOUČASNÉHO STAVU

V ročníku 2009/2010 mělo tento předmět zapsáno 69 studentů. V nejlepším možném případě je vyučujícím zasláno 69 x 6 zápočtových programů s dokumentací (tj. 414 prací), v případě nejhorším 69 x 7 x 2 prací (tj. 996 prací). Průměrný čas strávený opravou jedné práce je 20 minut a počet odevzdaných prací v průběhu semestru je přibližně 500, z čehož vyplývá, že čas strávený opravou prací je 166 hodin. Semináře tohoto předmětu jsou vedeny dvěma vyučujícími, takže průměrný čas strávený opravou přesahuje u každého vyučujícího 80 hodin.

2.1 PODMÍNKY ABSOLVOVÁNÍ

Podmínkou absolvování je odevzdání a splnění 7 zápočtových programů v určených termínech a splnění bodového limitu ze dvou zápočtových testů. Při uznání 7 odevzdaných zápočtových programů s dokumentací do stanoveného termínu a účasti na alespoň 8 seminářích je první, respektive druhý, zápočtový test uznán. Dále jsou při odevzdání a uznání 6 zápočtových programů do stanoveného mezního data, které se nachází před mezním termínem odevzdání, odpuštěny studentovi dva zbylé zápočtové programy. Zde se klade důraz na vlastní iniciativu studenta, kdy ke splnění předmětu nemusí odevzdat všech 8 příkladů, ale pouze 6. Nutnost odevzdání pouze 7 z 8 příkladů dává všem studentům možnost vyhnout se příkladu, který plně nepochopili a mohl by je demotivovat už na začátku studia tím, že nejsou schopni dobrat se k požadovanému výsledku. Ze zkušenosti vyučujícího takový student raději v prvních týdnem výuky přechází na jiný obor.

Tyto možnosti odevzdání menšího počtu příkladů jsou snahou k minimalizaci příkladů odevzdaných na poslední chvíli v mezním termínu. Velmi často se stává, že studenti čekají na poslední chvíli s odevzdáním a poté se vyučujícímu v systému objeví desítky odevzdaných příkladů najednou a to velmi prodlouží rychlé opravení všech prací a znejistí studenty, protože ti nejsou s dostatečným předstihem upozorněni na chyby, které se s vysokou pravděpodobností objeví v jejich dalších odevzdaných pracích.

Výběr příkladů probíhá na začátku semestru na úvodním semináři rozlosováním. Student si vylosuje právě jeden příklad z každé kategorie, který má možnost řešit buďto

na hodinách nebo samostatně. Součástí uznání příkladu je obhájení případných připomínek vyučujícího.

Práce jsou odevzdávány v elektronické formě přes portlet Zápis a odevzdávání prací, který je dostupný na Courseware předmětu. Příklad je vrácen k přepracování, pokud je chybný či nedostatečně propracovaný, a je možno ho odevzdat pouze jednou, ale s možností několikanásobné opravy. Limit oprav je u předmětu stanoven na 8, ale jeden příklad není limitován jednou opravou. Vrácený příklad může student přepracovat i vícekrát s tím, že každá oprava ho stojí jeden bod z limitu oprav prací. Opravy lze tedy v rámci série příkladů přenášet.

2.2 POUŽITÝ PROGRAMOVACÍ JAZYK

Zvoleným programovacím prostředím je Embarcadero Delphi 2009 od firmy Embarcadero Technologies pro operační systém Microsoft Windows. Jedná se o velmi vítanou změnu z pohledu studentů, protože velmi populární programovací jazyk Pascal od firmy Borland se potýkal s nepříznivým přístupem studentů v minulých letech, kdy byl označován za přežitek a neperspektivní programovací jazyk, který je málo využíván.

Borland Pascal byl dlouho dobu prvním programovacím jazykem, se kterým se student při studiu programování setkal. Vznikl s účelem vytvořit programovací jazyk, který je svou strukturou vhodný k výuce programování. Mezi jeho přednosti patří omezený počet jednoduchých instrukcí, které jsou pro začátečníka snadno srozumitelné a jejich funkce se dá odvodit z pouhého názvu. Ve své původní formě byl Pascal čistým procedurálním jazykem. Později přišla na trh verze Turbo Pascal 5.5, ve které byla firmou Borland přidána podpora objektově orientovaného programování, tzv. Object Pascal.

Object Pascal se později vyvinul v programovací jazyk Delphi zaměřený na platformu Microsoft Windows. Obsahuje integrované grafické prostředí a možnost přímého návrhu vizuálního vzhledu programu, což urychluje vývoj samotného programu. Kostra zdrojového kódu se automaticky vytváří na základě použitých ovládacích prvků.

Náhrada Pascalu jako výukového programovacího jazyka novějšími Delphi je možná díky programování konzolových aplikací. Zápis syntaxe programu je velmi podobný a výstupy z obou programů jsou bezmála totožné. Původní myšlenka výukového programu pro začátečníky je dodržena, ale zároveň obohacena o mnohem větší možnosti.

3 STUDENTSKÉ PRÁCE

Odevzdaná studentská práce pro předmět Programování 1 obsahuje několik předem daných částí (souborů). Jedná se o soubory se zdrojovým kódem a doprovodným textovým dokumentem, který popisuje řešení zadání.

3.1 STRUKTURA TEXTOVÉHO DOKUMENTU

K odevzdaným příkladům student vypracuje průvodní textový dokument ze šablony umístěné na Courseware předmětu. Dokument odevzdaný k příkladu musí obsahovat následující:

1. titulku (úvodní titulní stranu),
2. zadání příkladu,
3. stručný popis řešení (v souladu se zdrojovým kódem),
4. vývojový diagram či strukturogram odpovídající popisu řešení (je-li ze zadání vyžadován),
5. seznam jmen souborů se zdrojovým kódem (je-li jich víc, připojit i popis k čemu slouží),
6. ukázku programu (úvodní obrazovka se zadáním a výstupní obrazovka s řešením),
7. závěr (vlastní zhodnocení řešení daného příkladu).

Všechny výše uvedené sekce jsou začleněny do šablony dokumentace a jsou označeny záložkami, které jsou přípravou pro budoucí automatizaci opravy.

3.2 TÉMATA PRACÍ

Pro základní seznámení s programováním je připraveno několik kategorií příkladů. Tyto příklady jsou navrženy tak, aby studenty seznámily se základními operacemi a postupy, které jsou při řešení s programováním spojených problémů důležité. Všechna níže uvedená témata korespondují s příklady, které byly řešeny v programovacím jazyce Pascal v minulých letech, a jejich prostředí se sestává z konzolové aplikace velmi podobné konzolové aplikaci Pascalu.

Témata pro rok 2009/2010:

1. Vstup a práce s řetězci
2. Práce s maticí
3. Práce se záznamem
4. Typ množina
5. Práce s textovým souborem
6. Práce s obecným souborem
7. Předání parametru funkce či procedury odkazem a hodnotou
8. Řešení nekonečných řad

Probíraná látka patří mezi základní nutné znalosti programátora. Jedná se jak o práci s různými vstupy a výstupy - práce s řetězcem a souborem, tak o složitější matematické problémy jako je například řešení nekonečných řad. Schopnost pracovat s funkcemi a procedurami je v následujícím studiu nutností a student je již v jeho prvních programech směrován k využívání tohoto programovacího postupu. Při přechodu z Pascalu na Delphi je práce s funkcemi a procedurami o to více důležitá, protože Delphi do jisté míry automaticky generuje jejich strukturu a je zaměřeno především na objektově orientované programování.

4 SOUČASNÝ POSTUP OPRAVY ZÁPOČTOVÝCH PROGRAMŮ S DOKUMENTACÍ

4.1 SYSTÉM PŘIJÍMÁNÍ NOVÝCH PRACÍ A JEJICH EVIDENCE

Pro vyučujícího je prvním krokem při opravě přihlášení se do systému přes Portál ZČU a výběr položek Moje výuka, odevzdání prací. Zobrazí se aplikace Portálu ZČU Zápis a odevzdání prací. Zde vybere předmět, který bude opravovat, tj. Programování 1. Pokračuje na volbu Odevzdané práce, které jsou řazeny podle tématu a osobního čísla, což znamená, že lidé s příjmením ze začátku abecedy mají podle původního nastavení filtrování výhodu rychlejší opravy - vyšší priority. Dále se zvolí filtr Dosud neopraveno, aby se nezobrazovaly již opravené práce, a poté se výsledek seřadí podle data odevzdání, tématu a čísla pokusu, čímž se předejde zvýhodnění studentů s příjmením začínajícím písmenem, které je blíže začátku abecedy.

V současné době vyučuje semináře předmětu Programování 1 více vyučujících, takže je nutno přepínat mezi sekcemi jednotlivých vyučujících.

Soubory odevzdané do sekce speciální slouží hlavně pro opravu nezávažných chyb, kterých se student dopustil a jež by ho stály jeden z 16 pokusů odevzdání. Mezi tyto chyby patří např. odevzdání nekompletního souboru (chybí jeden z požadovaných souborů); je přiložena stará neopravená dokumentace; program funguje a všechny body zadání jsou splněny, ale v dokumentaci je chyba vícenásobného odřádkování pomocí klávesy Enter. Pokud si student svůj omyl uvědomí, tak je odevzdání stornováno a má možnost zaslat soubor s opravenou chybou do sekce speciální, na kterou se nevztahuje limit odevzdání.

Evidence rozlosovaných zadání, odevzdaných, neodevzdaných, uznaných, neuznaných příkladů a data jejich odevzdání se provádí pomocí dokumentu aplikace Microsoft Excel. Zkopírováním údajů z webového prostředí Portál ZČU se provede import studentů na začátku roku. Dokument podporuje jednoduché filtrování dle obsahu buněk (např. na skupiny A, B, C, D). Zápis vylosovaných předmětů probíhá na začátku roku po rozlosování jednotlivých příkladů na úvodních hodinách jednotlivých skupin. V případě, že si vyučující není jist tím, že příklad odevzdaný studentem koresponduje se zadáním ze začátku roku, je nutné otevřít jak dokument s evidencí, tak dokument se zadáními a porovnat ho se zadáním uvedené v dokumentaci odevzdané studentem. Dále dokument eviduje docházku studentů na cvičení předmětu, protože účast na 8 seminářích mu

promíjí nutnost napsat první zápočtový test. Všechny úkony prováděné s odevzdanými soubory se též zaznamenávají, ať už se jedná o stornované, odevzdané či uznané opravy.

Dokument Excel obsahuje přehled odevzdaných prací pomocí navržené funkce, jež barevně odlišuje odevzdané práce studenta - pouze pro přehlednost, a evidenci bonusů za docházku a splnění příkladů do určitého termínu. Tento dokument není přístupný studentům, takže si sami musí vést přehled o tom, jak jsou na tom s opravami příkladů.

Studentům jsou dále zveřejňovány výsledky testů a to přes stránky Katedry Výpočetní a Didaktické techniky (kvd.zcu.cz). V sekci studijní výsledky je umožněno studentům prohlédnout si výsledky testů přes zabezpečený přístup Orion konta.

4.2 OPRAVA ODEVZDANÉ PRÁCE

Pro opravu je nutné otevřít odeslanou práci a uložit její soubory na disk, kde se ukládají podle struktury, kterou vytváří sám systém. Systém umožňuje hromadné stažení prací na disk podle zadaných filtrovacích kritérií v .ZIP archivu. Tento archiv je postupně rozbalován do určené složky dle struktury systému. Struktura zabalených souborů navržená systémem má jednu nevýhodu, která znesnadňuje vyhledávání studentů a jejich prací, a to formát zápisu Studijní číslo + Jméno + Příjmení.

Do stejného adresáře, kam byl nakopírován obsah zabaleného souboru se studentovou prací, je přidán dokument aplikace Word s připomínkami k jednotlivým hodnoceným kategoriím, který je vytvářen ze šablony a odesílá se studentům, aby měli zpětnou odezvu k odevzdané práci.

Prvním krokem při opravě je kontrola odevzdaných souborů, tj. textový dokument, soubor se zdrojovým kódem a samotný program. Poté se zkontroluje zadání proti evidenci příkladů v Excel souboru, čímž se zjistí, jestli byl daný příklad už opraven, vrácen k přepracování, uznán nebo dosud nepřijat.

Při nové opravě je vhodné zkopírovat soubor s připomínkami z předchozí příkladu odevzdaného studentem. Výhoda je v tom, že student má tendenci dělat stejné chyby a tímto jednoduchým úkonem je možno sledovat jeho vývoj a možná zlepšení v postupu jeho práce, případně ho upozornit na stále se opakující chyby a možnost konzultací.

Jedním z kroků u opravy práce je ověření zadání studenta, zda koresponduje se zadáním vylosovaným na začátku školního roku. Zde zatím není jiná možnost než se podívat do Excel souboru s evidencí studentů a jejich prací a poté do souboru s očíslovanými zadáními.

Následuje kontrola funkčnosti programu s ohledem na zadání. Zde velmi záleží na konkrétním zadání, ale většina příkladů má pro přesně daný vstup přesně daný výstup. Bohužel je nutné mít tato typová řešení v externím souboru, což prodlouží opravu práce o čas nutný k vyhledání souboru a požadovaných vstupních a výstupních parametrů v něm.

Práce s programem při kontrole funkčnosti je také dobrým ukazatelem uživatelské přívětivosti, ovladatelnosti a případné kontroly chybných vstupů (neexistující cesta k souboru, konflikt datových typů atd.).

Pro kontrolu zdrojového kódu se používá jednoduchý textový editor se zvýrazněnou syntaxí (např. Notepad++). Když program nefunguje podle zadání a kontrola algoritmu programu neposkytuje jednoduché odhalení problému, tak je nutné otevřít zdrojový soubor v programovacím prostředí Embarcadero Delphi 2009 a provést úkony vedoucí ke zjištění chyby. Výstupem pro studenta jsou připomínky ohledně přehlednosti kódu, správnosti či nesprávnosti použitého řešení, odsazení kódu, formátování výstupů a další.

V textovém dokumentu se po provedení předchozích kroků kontroluje sekce Stručný popis řešení. Z této sekce by mělo být patrné, jakým způsobem byl program vyřešen a jestli navržené řešení koresponduje s řešením provedeným ve zdrojových souborech a samotném programu.

Následuje kontrola vývojového diagramu či strukturogramu, kde je opět důležité, aby navržený algoritmus odpovídal algoritmu v programu a byl přehledný a lehce čitelný. Podstatnou chybou v zápisu vývojových diagramů / strukturogramů je špatný zápis jednotlivých funkcí (chybně zapsaný cyklus). V připomínce je nutné na toto upozornit a odkázat studenty na materiály, které se touto problematikou zaobírají a pomohou studentovi chybu v budoucnu odstranit bez nutnosti vysvětlování každé chyby a demonstrace její opravy. Vývojové diagramy / strukturogramy jsou pouze u 4 z 8 příkladů,

protože některé programy mají složitou programovou složku. Vytvoření vývojového diagramu / strukturogramu pro takto složitý program by bylo pro studenty nad rámec úvodu do základů programování. Cílem je naučit studenty základnímu zápisu, který snadno pochopí.

Jednou z posledních kontrol dokumentu je kontrola grafické úpravy, stylu jazyka a formátování dokumentu. Zobrazením skrytých znaků v dokumentu se velmi snadno odhalují základní prohřešky proti správnému formátování dokumentu, např. nadbytečné Entery, vícenásobné mezery atd.

Hodnocení souladu se zdrojovým kódem se provádí po uznání dílčích součástí a to: zadání, samotného programu, stručného popisu řešení a případně vývojového diagramu / strukturogramu.

Závěr v dokumentaci studenta slouží pro získání informací o zadaném příkladu a vlastních připomínek studenta k němu.

Výsledky těchto kontrol jsou přeneseny do souboru s připomínkami, kde je nutné všechny připomínky ručně vypsat, aby se studentovi dostalo nejlepší možné zpětné odezvy, která v budoucnu eliminuje studentem prováděné chyby v návrhu programu. Připomínky se velmi často opakují s malými variacemi. V sekci Závěr jsou shrnuty informace pro studenta, např. možnost odeslání opravené práce do sekce speciální, uznání práce s připomínkami či neuznání práce a nutná oprava dle připomínek.

Po provedení všech oprav je výsledný soubor s připomínkami uložen do formátu PDF a odeslán studentovi, ve webovém prostředí Portálu ZČU se odeslanému souboru přiřadí jeden z příznaků - vrátit k přepracování nebo akceptovat**.** Bohužel není v možnostech Portál ZČU informovat uživatel (např. emailem) o opravení práce. Poslední akcí je doplnění údajů o opravě do souboru s evidencí příkladů.

5 HODNOCENÍ OPRAVOVANÝCH PRACÍ

Hodnocení zaslaného příkladu prochází několika fázemi, které mají vliv na uznání či neuznání. Cílem je nalezení chyb, jež student učinil při návrhu programu a dokumentace. Na tyto chyby je student upozorněn a je nasměrován na správné řešení, které by se mělo odrazit v dalších odevzdaných pracích, kde by se dříve dělané chyby už neměly opakovat.

5.1 PROGRAM

Hodnocení programu se provádí ze 3 základních hledisek - funkčnosti, ovladatelnosti a uživatelské přívětivosti. Největší vliv na splnění požadavků má funkčnost.

Funkčnost se dá otestovat zadáním vstupních hodnot a kontrolou hodnot výstupních. Některé programy jsou ve svém základu složitější a je tedy vhodné mít typová řešení pro každou variantu zadání. Práci usnadní dopředu připravený dokument či program se vstupními hodnotami a k nim korespondujícími výstupy. Problém, který je s vyhledáním a otevřením dokumentu / programu spojen, je zvýšení časové náročnosti opravy. Nejprve je nutné zjistit přesné zadání, které program studenta řeší, což vede k otevření dokumentace programu, kontrol správnosti zadání oproti evidenci příkladů, následnému nahlédnutí do souboru se zadáními příkladů a vyhledání typového řešení v dokumentu / programu. Vstupní hodnoty jsou voleny způsobem, který má odhalit případné chyby v algoritmu programu. Některé programy totiž mohou pracovat správně, pokud jsou zadané hodnoty v rámci určitých parametrů nastavených studentem, ale jakmile se tyto hodnoty dostanou mimo zvolené parametry (stále v souladu se zadáním příkladu), tak je program přerušen chybovým hlášením nebo je výsledek špatný.

Ovladatelnost a uživatelská přívětivost je hodnocena z pohledu vyučujícího jako celkový pohled na strukturu programového výstupu, který je uživateli dostupný v konzolové aplikaci. Důraz se klade zejména na přesné instrukce, které program klade uživateli a provádí ho postupně programem. Zpravidla je nutné, aby program přesně definoval co je vstupem, aby se předešlo zmatení při práci s programem. Dále je to přehledný výstup programu, ze kterého je patrné, co je vlastním řešením. Součástí ovladatelnosti a uživatelské přívětivosti je také kontrola chybných vstupů, na které by měl

být uživatel upozorněn a v nejlepším případě, pokud je to v silách studenta, vyzván k novému zadání.

Pro některé studenty je při prvním setkání s programováním poměrně těžké přemýšlet nad uživatelskou přívětivostí, zejména pokud mají velký problém s obecným pochopením programování samotného. V těchto případech bývají malé prohřešky proti přehlednosti a uživatelské přívětivosti pouze vytknuty v dokumentu s připomínkami, čímž student kvůli malé chybě, kterou si neuvědomil, neztrácí jeden pokus na odevzdání. Pokud se student v následujícím odevzdaném příkladu prohřeší stejným způsobem, tak je upozorněn na opakovanou chybu a příklad je neuznán s možností zaslání příkladu do sekce speciální. Jedná se o velmi dobrou metodu, kdy první neúmyslná chyba neznamená ztrátu pokusu, ale opakovaná nedbalost či nepozornost je penalizována nutností chybu odstranit a příklad znovu odevzdat.

5.2 ZDROJOVÝ KÓD

Kontrola zdrojového kódu probíhá u fungujícího programu vizuální kontrolou algoritmu, kvality řešení a použitého formátování. Důraz se klade na správné formátování, kvalitu kódu a zvolený postup. Nepřehledné formátování je vytýkáno a vyučující se snaží studenta nasměrovat připomínkami na přehlednější řešení. Kvalitou řešení je myšleno použití řešení, které je z programátorského hlediska efektivně implementováno.

Při vytváření složitějších programových celků je vhodné komentovat programové bloky a prvky, nejen kvůli usnadnění orientace v kódu, ale také kvůli zvýšení povědomí studenta o tom, co vlastně program v jednotlivých blocích a funkcích vykonává. Mnohokrát si tak student uvědomí, že místo opakování sady několika instrukcí je lepší zvolit funkci volanou parametrem či odkazem, která zkvalitní a zpřehlední výsledný kód.

Přehlednost zdrojového kódu je také nápomocná při špatné funkčnosti programu, kdy je následná editace za účelem opravení chyb/y o mnoho rychlejší, protože je snadnější se orientovat v okomentovaných a správně naformátovaných blocích programů. Vyučující tak může snadno nasměrovat studenta na požadovaný blok, který obsahuje chybu, aniž by musel explicitně identifikovat část kódu, ve které k chybě dochází. Student je tedy připomínkami motivován k tomu, aby se již ve svých začátcích programování zaměřil nejen na funkčnost, ale i na kvalitu kódu, i přestože kvalita kódu má v očích

studenta na funkčnost programu minimální vliv. Student se připravuje na budoucí povolání učitele výpočetní techniky a cílem je naučit studenta přehlednému zápisu, který bude předávat svým studentům.

5.3 DOKUMENTACE

Dokumentace je průvodním textem k samotnému programu a měla by vyučujícímu upřesnit použité algoritmy a postupy v programu. Klade se důraz na jazykové prostředky studenta, kterými svůj program prezentuje a popisuje. Student studuje technický obor a jím dodaná dokumentace by měla korespondovat s požadavky kladenými na budoucího učitele výpočetní techniky, proto jsou použité jazykové prostředky zahrnuty do hodnocení dokumentace.

Grafická úprava a formátování dokumentu čerpá z předem vytvořené šablony a je zde malý prostor pro vlastní úpravu, což je jednoznačné pozitivum z hlediska vytváření aplikace pro kontrolu studentských prací, která potom může s připravenou dokumentací pracovat a získávat z ní data na programové úrovni.

Soulad dokumentace s programem a se zdrojovým kódem je podmínkou pro splnění příkladu. Ke splnění tohoto kritéria je potřeba, aby funkce programu, zdrojový kód, stručný popis řešení a vývojový diagram / strukturogram odpovídal zadání studenta. Nejčastější chyby se objevují právě při zpracování vývojových diagramů / strukturogramů, kdy student špatně pochopí zápis či zamění formát cyklu, který v programu používá. Toto kritérium je tedy souhrnným ohodnocením studentovy celkové práce jak na programu, tak i na dokumentaci.

Závěr ve studentově dokumentaci slouží ke shrnutí práce a získání informací o pohledu a názoru studenta na zadání programu a případných problémech s vypracováním. Pro vyučujícího je tento, většinou krátký, odstavec neocenitelnou zpětnou vazbou na jím vypracovaná zadání. Pokud student vyjádří nespokojenost se zadáním, tak je mu dána možnost navrhnout lepší řešení nebo se podílet na změně zadání stávajícího. Komunikace při vypracovávání příkladů funguje oběma směry, jak ze strany vyučujícího pomocí připomínek zasílaných studentům, tak i ze strany studentů, kteří mají možnost se v závěru své práce vyjádřit k celkové práci.

6 KATEGORIZACE NEJČASTĚJŠÍCH CHYB

Předmět Programování 1 je pro mnohé studenty první zkušeností s programováním. Dává vyučujícímu možnost naučit studenty správné zásady formátování, zápisu programu a vedení doprovodné dokumentace, aniž by se musel zaměřovat na odnaučování špatně zažitých postupů z předchozího studia. Problém nastává u studentů, kteří se s programováním setkali na střední škole, a jejich úroveň se pohybuje od začátečníků po pokročilé. Student se znalostmi z předchozího studia se může dopouštět chyb, které pro něj v předchozím studiu chybami nebyly. Proto je důležité nejčastější chyby kategorizovat a upozorňovat na ně všechny studenty už od prvních seminářů. Chyby se dají rozdělit na tři základní kategorie, závažné, které mají velký vliv na výslednou práci a pochopení zadání, méně závažné, které mají většinou pouze nepatrný či kosmetický vliv na vypracování práce, a speciální, které vznikají nedbalostí studentů, ale na program vliv nemají.

6.1 CHYBY V PROGRAMU

Problémů při vytváření programu může nastat velké množství. Některé chyby jsou pro funkčnost programu vyloženě kritické, kdežto jiné nemají na funkčnost negativní dopad a jsou pouze činiteli nepřehlednosti a uživatelské nepřívětivosti.

6.1.1 ZÁVAŽNÉ

Funkčnost je prioritou každého programátora a program, který nejde bez úprav spustit je nepřijatelným řešením. Nefunkčnost je tedy velmi závažnou chybou, ale díky možnosti otestovat a spustit si vytvořený program doma či ve škole před odevzdáním z ní činí poměrně málo častou chybu. Bohužel i přes možnosti odhalení nefunkčnosti programu se může studentovi stát, že na otestování programu po změně ve zdrojovém kódu zapomene a odevzdá program nefunkční, aniž by o tom věděl.

Neméně závažnou chybou je fungování programu pouze s určitými parametry, které nemusí přímo kolidovat se zadáním, ale mohou ho jen kvůli usnadnění obcházet. Student se snaží co nejrychleji dojít k řešení a opomene testovat program i pro jiné vstupy, než jím zadané. Problém tak nastane, když program dostane na vstupu data, která nejsou v rozsahu parametrů, se kterými pracuje, a dojde k chybě, která vyvolá v lepším případě přerušení programu a v případě horším zcela špatný výstup. V případě, že se toto

stane, vyučujícímu nezbývá nic jiného, než celý program řádek po řádku projít a porovnat se stručným popisem řešení a vývojovým diagramem. Tyto chyby se velmi těžko odhalují a identifikují, protože je většinou provází nesoulad dokumentace s programem a zdrojovým kódem.

Další zásadní chybou je velká nepřehlednost programu, kdy není jasně patrné, co je vstupem a co je výstupem programu. Například se jedná o program, který po spuštění čeká na zadání proměnné, ale uživateli není jasně popsáno, jaká proměnná je očekávána. Ze strany uživatele zbývá metoda pokus / omyl nebo nahlédnutí do zdrojového kódu, což zvyšuje časovou náročnost opravy a je v rozporu se základy programování. Uživatel by měl vždy mít přehled o tom, co po něm program požaduje a jaký je výstup.

6.1.2 NEZÁVAŽNÉ

Nezávažnou chybou je například špatný formát výstupu a zavádějící instrukce programu. Tyto chyby nemají na chod programu velký vliv, ale mohou uživatele nechtěně mást a komplikovat zadávání vstupů a odečítání výstupů. Řešením je upozornit studenta na tuto chybu a nechat ho zamyslet se nad jím zadanými instrukcemi pro uživatele, aby se toto nechtěné zmatení příště minimalizovalo a program byl srozumitelnější.

6.1.3 SPECIÁLNÍ KATEGORIE

Tato kategorie se zabývá chybami, které jsou vyhodnoceny v závislosti na schopnostech a znalostech studenta. Student, který s programováním začíná, není nucen kontrolovat všechna chybová hlášení, protože pro něj je úspěchem fungování programu přesně podle zadání a s přesně definovanými parametry. Naopak pro studenta, který má s programováním mnohem větší zkušenosti a opomene kontrolu chybných vstupů, může být toto klasifikováno jako závažná programová chyba. Opomenutí kontroly vstupních parametrů se může zdát, po právu, pro začátečníka zbytečné a nemající vliv na funkčnost programu, zde je dobré studenta na chybu pouze upozornit a přimět ho, aby se v dalších příkladech této problematice více věnoval. U pokročilého studenta by se tento prohřešek promíjet neměl, pokud je v jeho silách a možnostech daný problém vyřešit a opomenutí řešení pramení pouze z nedbalosti či lenosti.

6.2 ZDROJOVÝ KÓD

Ve zdrojovém kódu je velmi nesnadné kategorizovat chyby na závažné a nezávažné. Pokud je program funkční podle zadání, tak se způsob zápisu těžko vytýká. Vysvětlit studentovi začátečníkovi, že je lepší zapisovat program přehledně podle určitých pravidel, je obtížné, protože studentův zápis je funkční a změna názvů a zarovnání bloků programu nepřinese jiný výsledek. Zde záleží na vyučujícím, jak chybu ohodnotí a zdali doporučí přeformátování programu s tím, že bude odeslán do sekce speciální či dá pouze výtku do dokumentu s připomínkami a u příští opravy bude špatné formátování považováno za závažnou chybu.

V každém případě je vhodné do připomínek umístit příklad správného formátování s odůvodněním, že úpravy ve stávajícím programu budou v budoucnosti o mnoho snazší. Níže je uveden příklad nevhodného pojmenování funkce a parametrů (Obrázek 1), vhodného pojmenování funkce s parametry v české (Obrázek 2) a anglické (Obrázek 3) jazykové variaci.

```
function funkce1(s1, t1: string): boolean;
var soub : textfile;
begin
      assign(soub, s1);
      rewrite(soub);
      writeln(soub,t1);
      closefile(soub);
end;
```

Obrázek 1 – Nepřehledný způsob zápisu funkce

```
function VytvorSouborSTextem(CestaSouboru, Txt: string): boolean;
var TextovySoubor: textfile;
begin
      Assign(TextovySoubor, CestaSouboru);
      Rewrite(TextovySoubor);
      Writeln(TextovySoubor,Txt);
      Closefile(TextovySoubor);
end;
```

Obrázek 2 – Přehledný zápis funkce v českém jazyce

```
function CreateFileWithText(FilePath, Txt: string): boolean;
var TxtFile: textfile;
begin
      Assign(TxtFile, FilePath);
      Rewrite(TxtFile);
      Writeln(TxtFile,Txt);
      Closefile(TxtFile);
end;
```

Obrázek 3 – Přehledný zápis funkce v anglickém jazyce

Jak je z obrázků patrné, tak vhodně zvolené pojmenování názvů funkcí a jimi volaných parametrů má velký vliv na samotnou čitelnost programu a zároveň nulový vliv na vykonanou funkci, všechny 3 příklady jsou zaměnitelné a fungují naprosto shodným způsobem. Bohužel nejsou témata v předmětu Programování 1 tak rozsáhlá, aby byly programy delší než 100 řádků a vyžadovaly od studentů většího rozmyslu při volbě názvů, který by jim usnadnily orientaci ve zdrojovém kódu.

Zdrojový kód se student od studenta velmi liší, nejen kvůli zažitým postupům jednotlivých studentů, ale také kvůli rozdílné úrovni studentů. Zkušenější student použije nepříliš známou funkci (Obrázek 4) v programovacím jazyce a student s menšími zkušenostmi vyřeší problém těžkopádným opisem složeným z mnoha dílčích a jednoduchých úkonů (Obrázek 5). Ani jedno z řešení není špatné, ale mnohem lépe vypadá jednořádkové a kompaktnější řešení, než dlouhý blok programu, suplující již existující funkci. Označit nevhodně zvolený postup za chybu není ideálním řešením, jedná se spíše o nevyužití všech dostupných prostředků nebo jejich špatného a nešikovného použití. Student, který použije nešikovné řešení, se tak jistě nevyhýbá a neobchází zadání práce, ale naopak se snaží řešení splnit v rámci svých dovedností a znalostí.

Pro názornost jsou uvedeny ukázky funkcí, ve kterých se ke stejnému řešení dochází rozdílným způsobem.

```
function PrevodTxtNaMalaPismena(VstupTxt: string): string;
var Vystup: string;
begin
      Vystup := LowerCase(VstupTxt);
      Result := Vystup;
end;
```

Obrázek 4 – Efektivnější řešení převodu velkých písmen na malá

```
function PrevodTxtNaMalaPismenaPresCase(VstupTxt: string): string;
var i: Integer;
    Vystup: string;
begin
      for i := 0 to Length(VstupTxt) do
        case VstupTxt[i] of
          'A': Vystup := Vystup + 'a';
          'B': Vystup := Vystup + 'b';
          'C': Vystup := Vystup + 'c';
          'D': Vystup := Vystup + 'd';
          'E': Vystup := Vystup + 'e';
          {...}
          'U': Vystup := Vystup + 'u';
          'V': Vystup := Vystup + 'v';
          'X': Vystup := Vystup + 'x';
          'Y': Vystup := Vystup + 'y';
          'Z': Vystup := Vystup + 'z';
          else
            Vystup := VstupTxt[i];
        end;
      Result := Vystup;

end;
```

Obrázek 5 – Řešení převodu velkých písmen na malá opisem funkce

Uvedené funkce mají za úkol převod velkých písmen v řetězci na písmena malá. V prvním případě je použita funkce LowerCase, ve druhém je opis této funkce pomocí strukturovaného příkazu Case, který převádí jednotlivé znaky v řetězci na jejich ekvivalent v písmu malém. První řešení je o mnoho elegantnější a přehlednější, druhé je bohužel méně přehledné, ale je z něj patrné, že s ním byla mnohem větší práce. Těžko tedy hodnotit správnost řešení, lze jen doporučit použití jiného příkazu a postupu do budoucna.

6.3 DOKUMENTACE

Dokumentace k odevzdaným příkladům je nedílnou součástí studentovy odvedené práce, která popisuje postup při řešení zadaného příkladu. Chyby prováděné při zpracování dokumentace nebývají nijak závažné a jsou dány, pro většinu studentů, tím, že je to jejich první setkání s technickou dokumentací. Pro usnadnění práce je studentům dostupná šablona, která obsahuje všechny styly a základní formátování. Většina chyb v dokumentaci vyplývá z nepozornosti studenta a jeho malých zkušeností s tvorbou technické dokumentace. Tyto zkušenosti se předmět snaží studentům předat a pozvednout povědomí o jejich důležitosti v navazujících ročnících studia.

Nejzávaznější chybou je nesoulad dokumentace s programem a zdrojovým kódem. Student navrhne program, ale popsaný postup a vývojový diagram / strukturogram neodpovídá zadání či samotnému programu. V tomto případě je obtížné zjistit, kde je chyba a proč se tak stalo. V předmětu se klade velký důraz na konzistenci vyvozených závěrů s materiály, které byly pro vytvoření dokumentace použity. Pokud se student této chyby dopustí, tak je příklad automaticky vyhodnocen jako nevyhovující a tudíž neuznaný.

Další velmi častou chybou je chybné formátování, které se dá odhalit pomocí funkce Zobrazit vše v textovém editoru Microsoft Word. Tato funkce zobrazí všechny použité znaky v dokumentu a vyučující má možnost identifikovat chyby ve formátování. Mezi tyto chyby patří nadbytečné užívání odřádkování pomocí klávesy Enter. Tento nešvar vzniká kvůli systému postupu studentů při vytváření dokumentu. Student plní bod po bodu a potřebné formátování a oddělení bloků textu provádí pomocí nadbytečného řádkování. Tento postup je při vytváření rychlejší a efektivnější pro studenta pouze do chvíle, než je student nucen dokumentaci přepracovat a zjistí, že přidání textu do jednoho bloku mu posune celý zbytek dokumentu a je nucen zbytečné řádkování odmazat, aby zachoval čitelnou strukturu. I přestože se jedná o velmi častou chybu, kterou udělá většina studentů, je po několika upozorněních v dokumentu s připomínkami tento nešvar eliminován. Student si sám uvědomí, že používání stylů s mezerou před či za odstavcem je mnohem efektivnější pro případnou editaci, a rychle se adaptuje a přejímá efektivnější řešení.

Frekventovanou chybou je také špatný formát obrázků, které student používá v sekci Ukázka programu pro demonstraci funkčnosti programu. Obrázek ve formátu, který používá ztrátovou kompresi, může být velmi špatně čitelný kvůli artefaktům, které tato komprese způsobuje. Studentům se jako řešení doporučuje použití bezztrátové komprese, která artefaktům předejde.

Obrázky v dokumentaci nejsou až tak důležité pro vyučujícího, jako spíše pro studenta, kterému zaznamenání funkce programu pomůže představit si, jak vlastně jeho program vypadá, jak je přehledný a uživatelsky přívětivý. V dokumentaci je potřeba mít obrazovku se vstupem a obrazovku s výstupem. Tyto dva obrázky podávají ucelený pohled na program a větší množství obrázků popisujících práci s jednoduchým programem vede

vyučujícího k tomu, že program není tak uživatelsky přívětivý a snadno ovladatelný, jak se na první pohled zdá.

Nejčastěji se vyskytující chybou jsou u opravovaných příkladů neaktualizované informace. Chyba je opravena, ale dokumentace tuto změnu neodráží například ve vývojovém diagramu / strukturogramu nebo v postupu řešení. Tyto údaje mohou zpětně vést k nejzávaznější chybě a tou je nesoulad dokumentace s programem a zdrojovým kódem. Tato chyba u opravy je poměrně častá a student si ji většinou sám uvědomí a opraví, v tomto případě je možno opravu práce stornovat a umožnit studentovi poslání práce do sekce speciální, kde o pokus opravy nepřijde.

7 MOŽNOSTI AUTOMATICKÉ VALIDACE SEMESTRÁLNÍCH PRACÍ

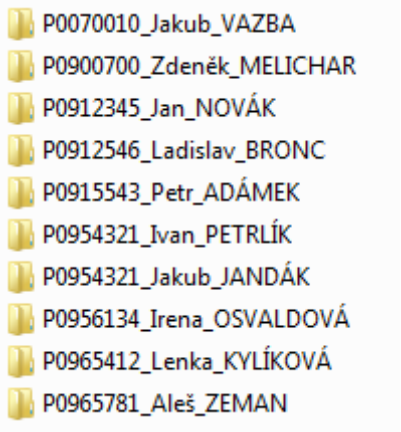
Pro předmět Programování 1 jsou možnosti automatické validace velmi široké. Je to dáno unifikovanou strukturou šablon pro vytvoření doprovodné dokumentace a zároveň nutností provádět spoustu dílčích a opakujících se úkonů spojených s vedením evidence a opravou prací. V kapitole 4 Současný postup opravy zápočtových programů s dokumentací je rozebrán aktuální stav průběhu oprav. Je použito několika dokumentů a programů, které tvoří funkční celek, ale tento celek není uživatelsky přívětivý a jeho správa je zdrojem časových ztrát. Cílem automatizace validace by měla být centralizace těchto dílčích prvků do jednoho přehledného programu a eliminace nadbytečných a opakujících se základních činností.

7.1 AUTOMATICKÁ EVIDENCE

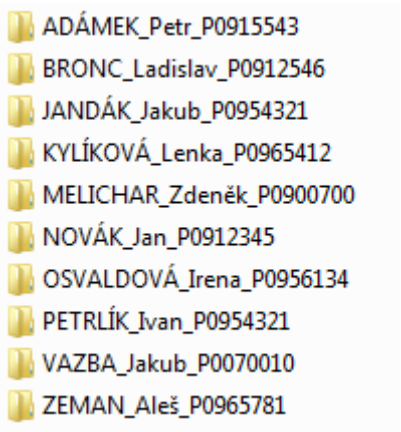
Dosud prováděná evidence se spoléhá na provázání dokumentu Excel a dalších dílčích dokumentů a programů (např. dokument se zadáními, dokumenty a programy s typovými řešeními příkladů). Tento postup v sobě zahrnuje velké množství opakujících se činností, které vedou k prodloužení času stráveného opravou.

Řešením je použití automatické evidence pomocí programu. Všechny potřebné údaje o přijatém příkladu jsou ukládány v prostředí Portál ZČU do CSV a ZIP souboru, který má jednoduchou vnitřní strukturu a snadno se s ním pracuje.

Po stažení souborů z Portálu ZČU se dá programově zařídit, aby byly ze souboru ZIP práce automaticky rozbaleny a roztřízeny do vhodnější adresářové struktury. Přehlednější strukturou než nyní používanou (Studijní číslo + Jméno + Příjmení) je struktura, kde na prvním místě figuruje Příjmení, dále Jméno a poslední Studijní číslo.



Obrázek 6 – Původní struktura evidence studentů



Obrázek 7 – Nově navržená struktura evidence studentů

Z obrázků je patrné, že rozdíl ve strukturách pojmenování je značný. Vyučující si u studenta zapamatuje spíše příjmení než jeho osobní číslo. Tato struktura nemá vliv na funkci programu. Práci by s adresáři prováděl program, ale pro případ nenadálé události, kdy je potřeba nahlédnout do databáze, je tato struktura vhodnější a pro vyučujícího přehlednější.

Navržená adresářová struktura se dá na začátku semestru vygenerovat z CSV souboru staženého z webového prostředí Portál ZČU. Jedná se o jednoduché rozdělení textového souboru a vytvoření adresářové struktury ze získaných dat. Vytvoření kompletní adresářové struktury pro semestr poté zabere pár vteřin, což je v porovnání s postupným ručním vytvářením databáze značná časová úspora.

Program by měl obsahovat vnitřní seznam studentů se základními informacemi, aby nebylo nutné opakovaně načítat vstupní CSV soubor a aby bylo možné provádět

základní editační úkony spojené s vedením databáze jako přidávání a mazání studentů (například z důvodu přechodu na jiný obor).

Odevzdané příklady přes Portál ZČU se dají stáhnout ve formátu ZIP a jejich roztřídění do jednotlivých složek se dá zařídit programově. Stačí metoda, která obsah souboru postupně rozbalí do příslušných adresářů. Aby vše proběhlo úspěšně, je nutné, aby byl formát staženého ZIP souboru pevně stanoven. Adresář s příklady studenta by měl obsahovat identifikační klíč, podle kterého se budou dané příklady třídit a ukládat na vhodné místo. Vhodným klíčem je studijní číslo, které je pro každého studenta unikátní a umožní zjistit do kterého adresáře a ke kterému studentovi soubor patří. Podle zkratky příkladu se vytvoří záznam v databázi odevzdaných prací, se kterým program může jednoduše pracovat. Použití příjmení jako klíče není vhodné z důvodu možnosti stejného příjmení více studentů. Zde by nastal problém se správným přiřazením příkladu ke správnému studentovi.

Pro evidenci prací by bylo vhodné mít v programu další databázovou strukturu, do které by se ukládaly informace z CSV souboru (datum odeslání souboru pro kontrolu dodržení termínů odevzdání), informace o cestách k jednotlivým souborům a příznaky příkladů (např. opraveno, uznáno, stornováno). Tato další databáze by byla provázána s databází studentů vytvořenou při založení nového projektu na začátku semestru pomocí unikátního klíče – studijního čísla.

Kombinace těchto dvou databází podstatně zkracuje dobu potřebnou pro vedení evidence studentů. Data přijetí není nutné ručně kopírovat z CSV souboru, ale program sám si tato data najde a přiřadí k odevzdanému příkladu automaticky při importování nových příkladů. Další výhodou je možnost filtrování příkladů podle určitých kritérií načtených z příznaků v databázi.

Programová evidence dle navržené struktury eliminuje většinu dílčích činností, které jsou nutné pro přehledné vedení databáze. Její struktura odpovídá dosavadní struktuře evidence prací, ale díky automatizaci činností, jako je ruční zjišťování a zápis lehce dostupných údajů ze souborů do databáze, a provázání jednotlivých bloků do jednoho programu je patrná mnohem menší časová náročnost.

7.2 AUTOMATICKÁ VALIDACE

Evidence cest k jednotlivým souborům a příznaků v databázi se dá provázat s validací v dalším bloku programu, který by mohl přistupovat k jednotlivým souborům daného příkladu, ze kterých by byla čerpána data pro vyučujícího ve sjednoceném formátu bez nutnosti ručně vyhledávat a spouštět dané soubory.

Za předpokladu, že výše zmíněné databáze fungují a jsou správně provázány, by bylo možné stáhnout z webového prostředí Portál ZČU pouze 2 soubory (ZIP obsahující studentské práce a CSV obsahující informace o odevzdaných souborech), které by byly při každé opravě automaticky zpracované. Tímto odpadne celá ruční evidence a je možné se věnovat pouze opravě prací bez nutnosti evidovat jednotlivé kroky a ověřovat podmínky plnění zápočtu jako za aktuálního stavu.

Přístup k souborům se dá provádět uvnitř programu a to pomocí několika metod. Moderní programovací jazyky podporují práci s OLE (Object Linking and Embedding) komponentou, která umožňuje přístup k Microsoft Office dokumentům na programové úrovni. Načítání doprovodné dokumentace a programových souborů je tedy o mnoho rychlejší, protože odpadá jakékoliv procházení adresářů, které obsahují jednotlivé soubory. Na každý z odevzdaných souborů by byl v programu odkaz, kterým by bylo možné daný soubor spustit.

Tuto funkci podporuje programovací prostředí Embarcadero Delphi. Toto prostředí je vhodným kandidátem na vytvoření takto sofistikovaného programu pro automatickou evidenci a validaci studentských prací. Delphi obsahuje v základu velmi mnoho knihoven a komponent, které usnadňují práci s databázemi, soubory a dokumenty. Díky obrovské komunitě a mnohaleté popularitě je možné na internetu najít všechny potřebné knihovny. Delphi v základu chybí knihovna či komponenta, která by umožnila rozbalení souboru ZIP. Tato knihovna je ovšem volně dostupná ve freeware licenci a její instalace spočívá v nahrání knihovny do adresáře projektu. Poté stačí v hlavičce programu připsat tuto použitou knihovnu a pomocí dvou příkazů rozbalit obsah adresáře do požadované složky.

Oprava práce by tedy byla rozdělena do jednotlivých kroků, které by se postupně zabývaly programem, zdrojovým kódem a dokumentací. Potřebná data by se čerpala z propojených databází a souborů uložených na pevném disku ve stanovené struktuře.

Časově náročnou činností při opravě práce je psaní připomínek, které by měly vystihnout, jaký problém v odevzdané práci je a jak jej napravit. Připomínky se velice často opakují, protože se studenti dopouštějí stejných chyb. Řešením by mohla být v programu integrovaná databáze nejčastějších připomínek u jednotlivých sekcí. Vyučující by v programu postupně procházel mezi jednotlivými sekcemi a spouštěl odpovídající soubory.

Posledními kroky je tedy shrnutí připomínek, vygenerování výstupního souboru s připomínkami a přiřazení příznaků do databáze prací dle stavu opravy – splněno, nesplněno, sekce speciální, stornováno. Výsledný soubor s hodnocením studentovi práce by se po opravě uložil na pevný disk a cesta k němu by byla uložena do databáze, ze které by byl soubor dostupný pro případnou zpětnou kontrolu. Soubor se odešle studentovi a práce je opravena.

7.3 MOŽNOSTI ODHALENÍ PODVODŮ

Momentálně je při pochybnostech vyučujícího jedinou možností si studenta pozvat do konzultačních hodin a několika otázkami zjistit, jestli program a dokumentaci vytvořil samostatně. Každý ze studentů použil pro vytvoření programu nějaký algoritmus a měl by být schopen ho popsat vlastními slovy. Naštěstí není tato praxe běžná a studenti, kteří v oboru Informatika se zaměřením na vzdělávání zůstanou, neinklinují k podvodům při řešení zadaných prací.

Automatická možnost odhalení podvodů by spočívala v průběžném vytváření databáze opravených a uznaných prací. Při načtení nových prací by se na pozadí programu spustilo vlákno, které by jednotlivé práce porovnávalo s databází.

Způsob, jakým by byly práce porovnávány, je velmi náročný a neefektivní. Z odevzdané dokumentace by se vybralo několik klíčových slov, která by byla porovnávána oproti databázi prací. Cílem je odhalit kopírovaný text, který může být i mírně pozměněn. Omezení počtu klíčových slov je nutné z důvodu velkého počtu odevzdaných prací, kdy by kontrola výskytu všech slov oproti databázi trvala nepředstavitelně dlouho. Pro porovnání

je možné získat procentuální shodu práce a v případě vysoké shody provést další průchod. Zde je pak na vyučujícím, aby práce osobně zkontroloval a porovnal.

Odhalení podvodů ve zdrojovém kódu je velmi podobné způsobu odhalování podvodů v dokumentaci s mírnou úpravou. Odevzdané a schválené zdrojové kódy by byly umístěny do databáze, která by obsahovala několik údajů. Jedním z údajů by byly použité proměnné a jejich počet a typ, které se dají načíst ze sekce, kde se proměnné deklarují. Při přijetí nového příkladu se dá velmi rychle spočítat počet proměnných určitého typu a tento výsledek se poté dá porovnat s databází. Tato metoda vychází z předpokladu, že student, který programování nerozumí a pouze kopíruje zdrojové kódy, často jen změní názvy proměnných a upraví formátování programu. Kontrola formátu a rozčlenění programu je tedy jako možný způsob odhalení podvodu eliminován tímto vcelku jednoduchým úkonem. Naopak zásah do algoritmu programu je pro takového studenta velmi obtížnou činností, která vyžaduje určitý stupeň znalostí. Přejmenování proměnných nemá na tuto metodu vliv, neboť se nekontroluje jejich název, ale pouze četnost výskytu a typ. Vyučujícímu by se opět zobrazila procentuální shoda zdrojových kódů a opět je na něm, aby provedl hlubší kontrolu.

Automatické odhalení podvodů má za dosavadního stavu velmi malou prioritu. Studentů na oboru Informatika ve vzdělávání spíše ubývá poté, co se setkají s předmětem Programování 1, a tím je pro vyučujícího mnohem snazší věnovat se studentům na individuálnější úrovni, což snižuje možnosti studentů okopírovat práci spolužáka a ztratit se ve velkém počtu opakujících se algoritmů a postupů v programech.

Odhalení podvodů tedy není možné plně zautomatizovat podle navrženého modelu, je pouze možné spočítat procentuální shodu prací a zdrojových kódů a výsledky zobrazit vyučujícímu, který se může na porovnání prací zaměřit a posoudit zda se jedná o podvod. Časová náročnost těchto metod se dá snížit tím, že práce, které se kontrolují, budou porovnávány oproti pracím ze stejného tématu. V případě 8 témat by se přijatá práce porovnávala pouze s 1/8 prací.

8 POPIS OVLÁDÁNÍ A FUNKCE PROGRAMU

Program je naprogramován v prostředí Embarcadero Delphi 2010 a je určen pro platformu Microsoft Windows. Funkčnost programu se opírá o programy Poznámkový blok a Microsoft Office Word.

Základem programu jsou 2 databáze a to databáze se seznamem studentů a databáze se seznamem prací. Obě databáze lze do jisté míry editovat, ale program je navržen tak, aby byla potřeba manuální editace minimalizována a prováděna pouze v nejnutnějších případech.

Pro efektivní práci s programem jsou potřebné 3 soubory z webového prostředí Portál ZČU – soubor ve formátu CSV se seznamem studentů, soubor ZIP a doprovodný CSV se studentskými pracemi. Soubor se seznamem studentů se načte na začátku semestru a dojde tím k vytvoření databáze jednotlivých studentů. Soubory obsahující studentské práce a informace o nich se importují pokaždé, když chce uživatel aktualizovat databázi se seznamem odevzdaných prací. Všechny následující panely sdílejí hlavní nabídku umístěnou v horní části programového okna, tato nabídka slouží k přecházení mezi jednotlivými sekcemi programu.

8.1 POPIS JEDNOTLIVÝCH PANELŮ

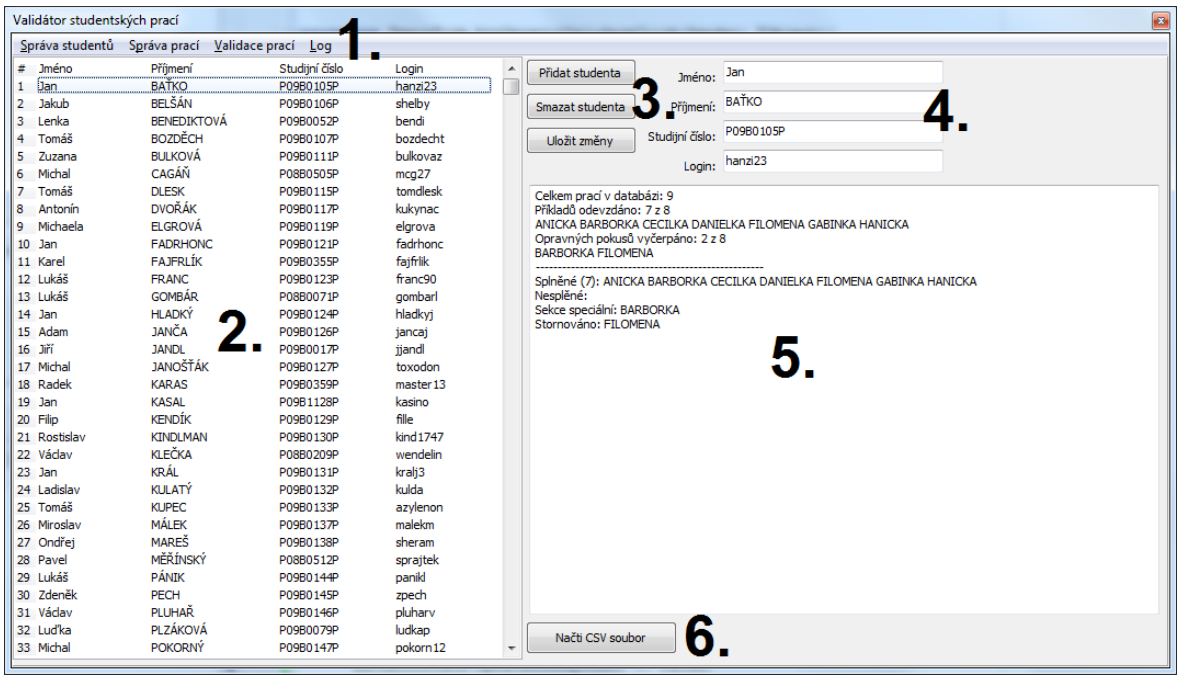
Program se skládá ze 4 základních částí, které spolu navzájem komunikují a umožňují zrychlit validaci studentských prací. Jedná se o části: Správa studentů, Správa prací, Validace prací a Log.

8.1.1 SPRÁVA STUDENTŮ

Výchozí panel, který se zobrazí po spuštění aplikace. Panel slouží k základní správě databáze studentů. Je z něj možno filtrovat práce zvoleného studenta a získat informace o již odevzdaných pracích. Pro zobrazení informací je využito obou databází – databáze studentů a databáze odevzdaných prací.

V levé části je komponenta typu String Grid, která zobrazuje informace načtené z databáze. V prostřední části jsou 3 tlačítka, která umožňují základní práci s databází, v pravé horní části se nachází 4 editovatelná pole, pravá dolní část obsahuje textovou

komponentu zobrazující stav odevzdaných prací a v dolní části je tlačítko sloužící pro import seznamu studentů z CSV souboru.



Obrázek 8 - Panel Správa studentů

1. Hlavní nabídka – umožňuje přecházet mezi jednotlivými částmi programu a je shodná pro každý z panelů
2. String Grid komponenta s aktuálním seznamem studentů
    a. zobrazuje aktuální seznam studentů
    b. po kliknutí na studenta načte informace do skupiny editovatelných polí (4.) a zobrazí informace o odevzdaných pracích daného studenta textové komponentě (5.)
    c. dvojklik na studenta vyfiltruje aktuální práce dle studijního čísla vybraného studenta a přepne na panel Správa prací, kde jsou práce zobrazeny (Obrázek 10)
3. Skupina editovacích tlačítek
    a. přidat studenta – přidá studenta do databáze s údaji uvedenými v editovatelných polích (4.)

b. smazat studenta – smaže aktuálně vybraného studenta z databáze

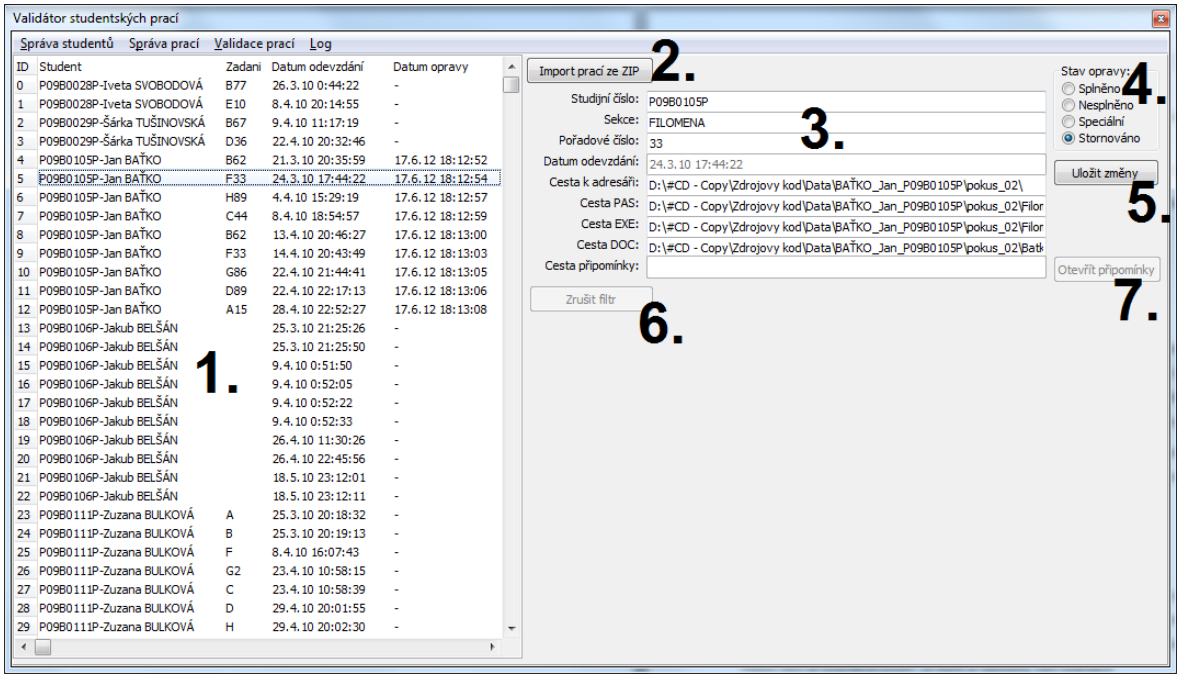
c. uložit změny – uloží změny provedené v editovatelných polích u aktuálně vybraného studenta

4. Editovatelná pole - načítají se do nich informace o vybraném studentovi a umožňují zpětnou úpravu údajů
5. Textová komponenta – shrnuje informace o pracích odevzdaných studentem
6. Tlačítko pro import seznamu studentů – importuje studenty do databáze z CSV souboru

8.1.2 SPRÁVA PRACÍ

Tento panel zobrazuje databázi odevzdaných prací a to ve dvou možných stavech. Stavem prvním je zobrazení celé databáze (Obrázek 9) a stavem druhým je zobrazení prací vyfiltrovaných dle studijního čísla (Obrázek 10). Filtrování dle studijního čísla se provádí dvojklikem na záznam v panelu Správa studentů. Zobrazení panelu je možné provést vybrání položky Správa prací z hlavního menu v horní části programového okna, případně je panel zobrazen ve chvíli, kdy dojde z panelu Správa studentů k filtrování prací.

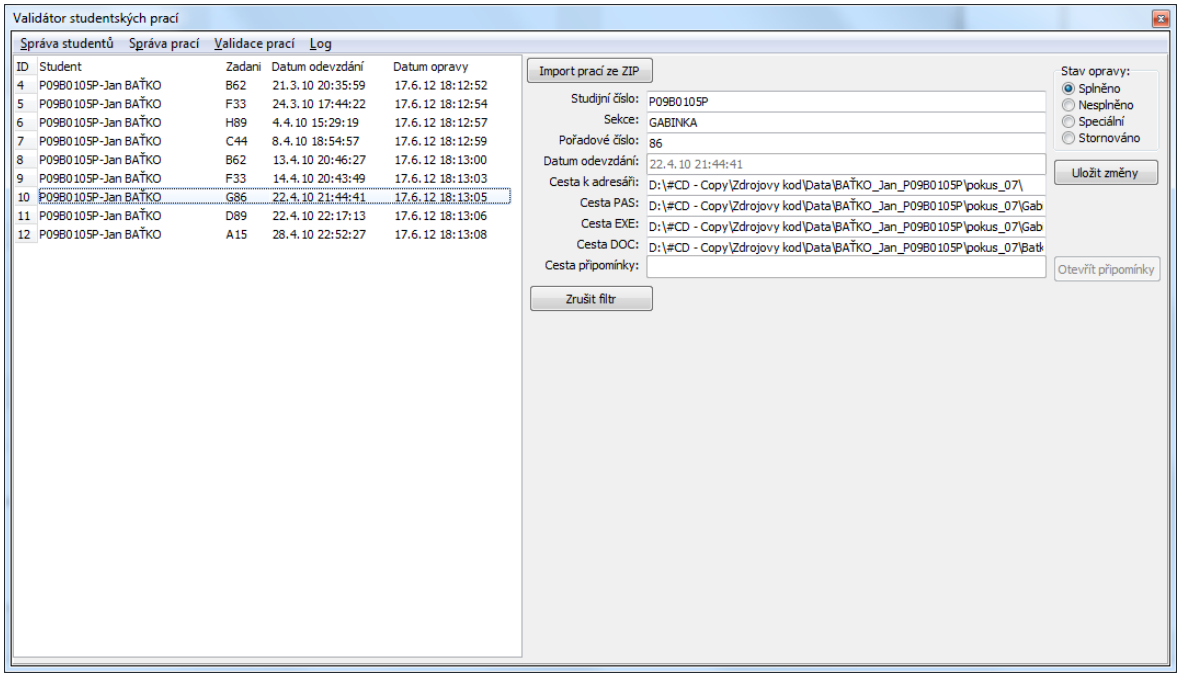
Komponentami panelu je v levé části String Grid s informacemi o databázi prací, v prostřední horní části se nachází tlačítko importu prací ze ZIP souboru, pod ním jsou editovatelná pole s informacemi o vybrané práci s tlačítkem pro zrušení filtrování, v pravé horní části je to Radio Group sloužící pro nastavení stavu práce, pod ním tlačítko pro uložení provedených změn nad vybranou prací a pod ním je poslední tlačítko, které otevře soubor s připomínkami v případě, že jsou připomínky dostupné.



Obrázek 9 - Panel Správa prací (zobrazení bez aplikovaného filtru)

1. String Grid komponenta s aktuálním seznamem prací
    a. zobrazuje seznam prací ve dvou stavech – bez aplikovaného filtru a s filtrem dle studijního čísla
    b. kliknutí na práci provede načtení informací o práci do skupiny editovatelných polí (3.)
    c. dvojklik na vybranou práci přepne na panel Validace prací, kde je možno vybranou práci opravit a okomentovat
2. Tlačítko pro import nových prací – import nových prací vyžaduje přítomnost dvou souborů (CSV a ZIP) s pracemi studentů ve stejné složce
3. Editovatelná pole – tato pole obsahují údaje o vybrané práci a taktéž slouží k jejich editaci
4. Radio Group s nastavením stavu opravy – zobrazuje a zároveň umožňuje nastavit stav opravy vybrané práce
5. Tlačítko sloužící k uložení provedených změn nad vybranou prací
6. Tlačítko umožňující zrušit aktuální filtr – tlačítko je aktivní pouze v případě, že je na seznam zobrazených prací aplikován filtr

7. Tlačítko pro otevření souboru s připomínkami – slouží k otevření souboru s připomínkami, pokud je daný soubor dostupný

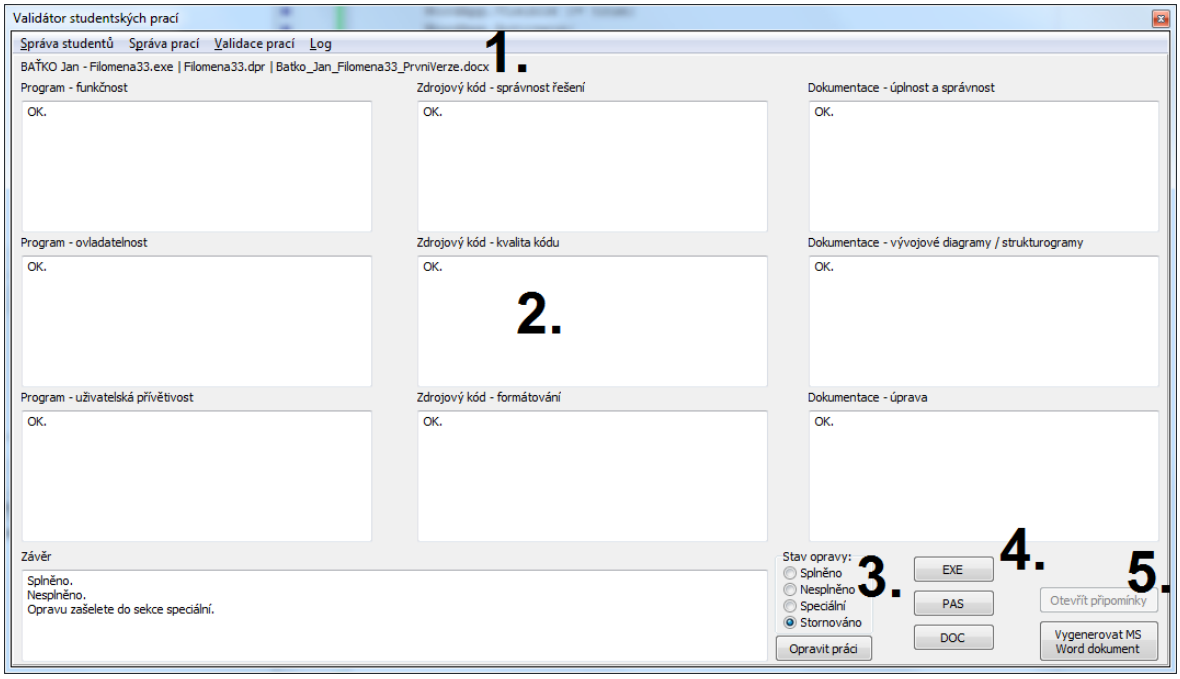


Obrázek 10 - Panel Správa prací (filtrované zobrazení)

8.1.3 VALIDACE PRACÍ

Funkční panel programu, který slouží k samotné opravě zvolené práce. Opravovaná práce je předána z panelu Správa prací a ovládací prvky na tomto panelu se odkazují do databáze odevzdaných prací.

Horní část obsahuje informace o vybrané práci, prostřední část obsahuje skupinu 10 textových polí, která slouží pro zapsání připomínek a pravá dolní část obsahuje Radio Group komponentu s tlačítkem sloužícím pro nastavení stavu opravy, dále skupinu 3 tlačítek, které otevírají korespondující soubory a poslední skupina dvou tlačítek umožňuje vygenerovat a případně otevřít dříve vygenerované připomínky.



Obrázek 11 - Panel Validace prací

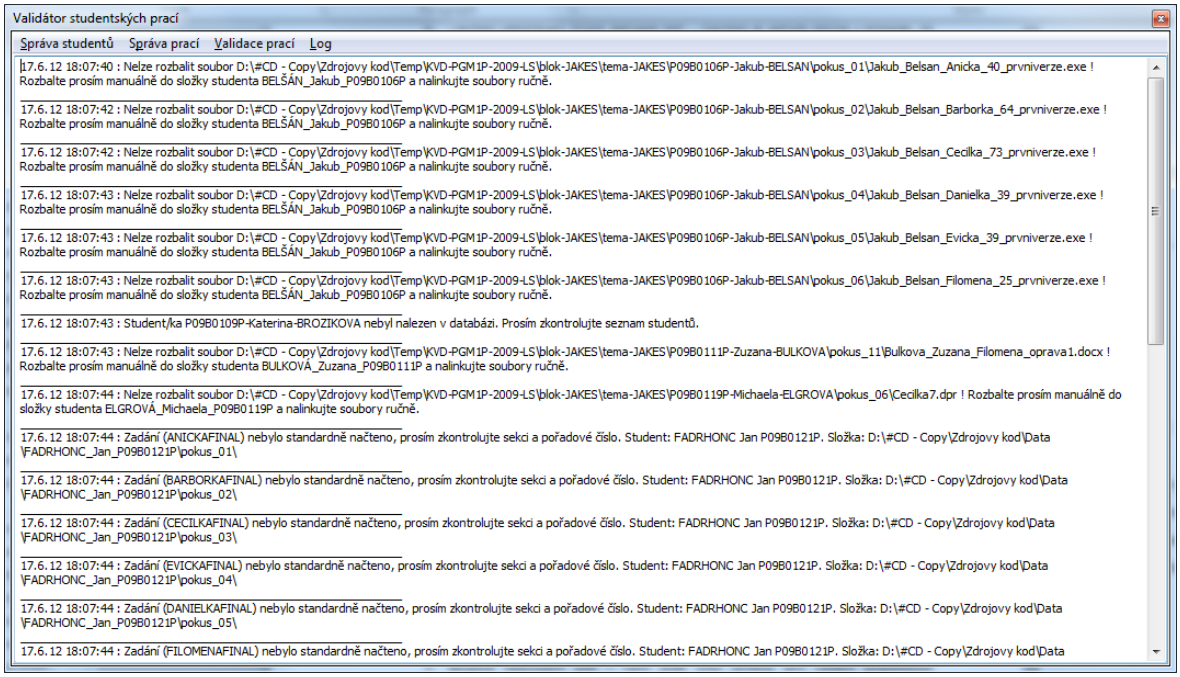
1. Informace o opravované práci
2. Skupina textových polí
    a. tato pole jsou určena pro zadání připomínek k jednotlivým částem opravované práce a jsou použita pro export připomínek do Microsoft Word dokumentu
    b. pole jsou ve zdrojovém kódu seřazena za sebou, takže je možno mezi nimi pomocí klávesových zkratek TAB a SHIFT + TAB přepínat
3. Radio Group s nastavením stavu opravy a tlačítko pro opravu práce – zobrazuje a zároveň umožňuje nastavit stav opravy vybrané práce, uložení nového nastavení opravy se provádí pomocí tlačítka Opravit práci
4. Skupina 3 tlačítek s odkazy na soubory odevzdané práce – pokud jsou odevzdané soubory dostupné, tak jednotlivá tlačítka otevírají korespondující soubory bez nutnosti procházení adresářové struktury manuálně
5. Tlačítka pro správu připomínek
    a. otevřít připomínku – tlačítko je aktivní v případě, že práce již byla jednou opravena a na disku se tedy nachází předchozí verze

připomínkového souboru, po kliknutí na aktivní tlačítko se otevře soubor s připomínkami

b. vygenerovat MS Word dokument – vygeneruje dokument s připomínkami, připomínky jsou převzaty z odpovídajících textových polí

8.1.4 LOG

Poslední panel aplikace, který obsahuje pouze jedno textové pole. Toto textové pole je napojeno na soubor v datovém adresáři programu a účelem je poskytnout historii informací o chybových hlášeních programu. Pokud program narazí na nestandardní chování při zpracovávání dat, je vyvolána událost, která danou chybu okamžitě zobrazí ve formě informačního okna a zároveň provede záznam do textového souboru na disku.



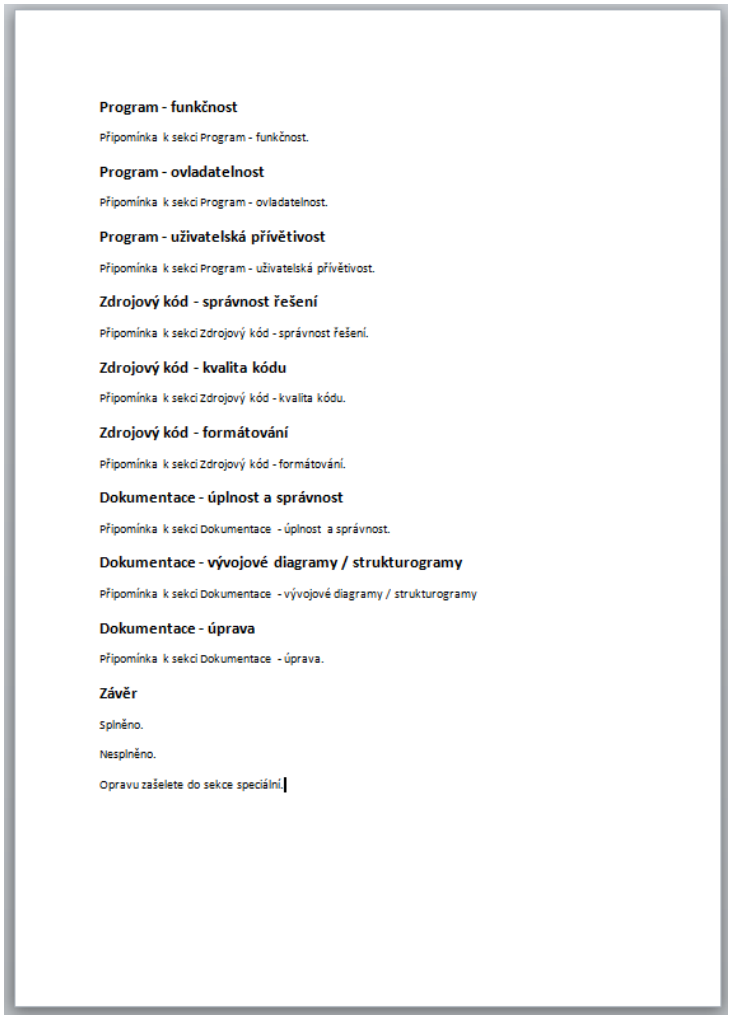
Obrázek 12 - Panel Log

8.2 PRÁCE S PROGRAMEM

Na začátku semestru se provede import studentů na panelu Správa studentů (manuálně nebo z CSV souboru z webového prostředí Portál ZČU) a dle potřeby se upraví. Odevzdané práce na webovém portálu je nutno stáhnout v souhrnném ZIP souboru včetně doprovodného CSV souboru a pomocí tlačítka Import prací ze ZIP na panelu Správa prací se provede import do databáze. Pokud se u dané práce import nezdaří, uživatel je na

tuto událost upozorněn informačním oknem a zároveň záznamem do logovacího souboru (a tím i do textového pole na panelu Log).

Informace o odevzdaných pracích je možno upravit na panelu Správa prací pomocí dostupných ovládacích prvků. Oprava práce se provádí dvojklikem na vybranou položku v panelu Správa prací – tato akce přesune uživatele na panel Validace prací, kde může využít předem připravená textová pole a tlačítka s rychlým přístupem k jednotlivým souborům. Po napsání připomínek se stisknutím tlačítka Vygenerovat MS Word dokument vygeneruje dokument, který je možno rovnou z prostředí aplikace Microsoft Word poslat studentovi. Posledním krokem opravy práce je nastavení a uložení příznaku se stavem opravy.



Program - funkčnost

Připomínka k sekci Program - funkčnost.

Program - ovladatelnost

Připomínka k sekci Program - ovladatelnost.

Program - uživatelská přívětivost

Připomínka k sekci Program - uživatelská přívětivost.

Zdrojový kód - správnost řešení

Připomínka k sekci Zdrojový kód - správnost řešení.

Zdrojový kód - kvalita kódu

Připomínka k sekci Zdrojový kód - kvalita kódu.

Zdrojový kód - formátování

Připomínka k sekci Zdrojový kód - formátování.

Dokumentace - úplnost a správnost

Připomínka k sekci Dokumentace - úplnost a správnost.

Dokumentace - vývojové diagramy / strukturogramy

Připomínka k sekci Dokumentace - vývojové diagramy / strukturogramy

Dokumentace - úprava

Připomínka k sekci Dokumentace - úprava.

Závěr

Splněno.

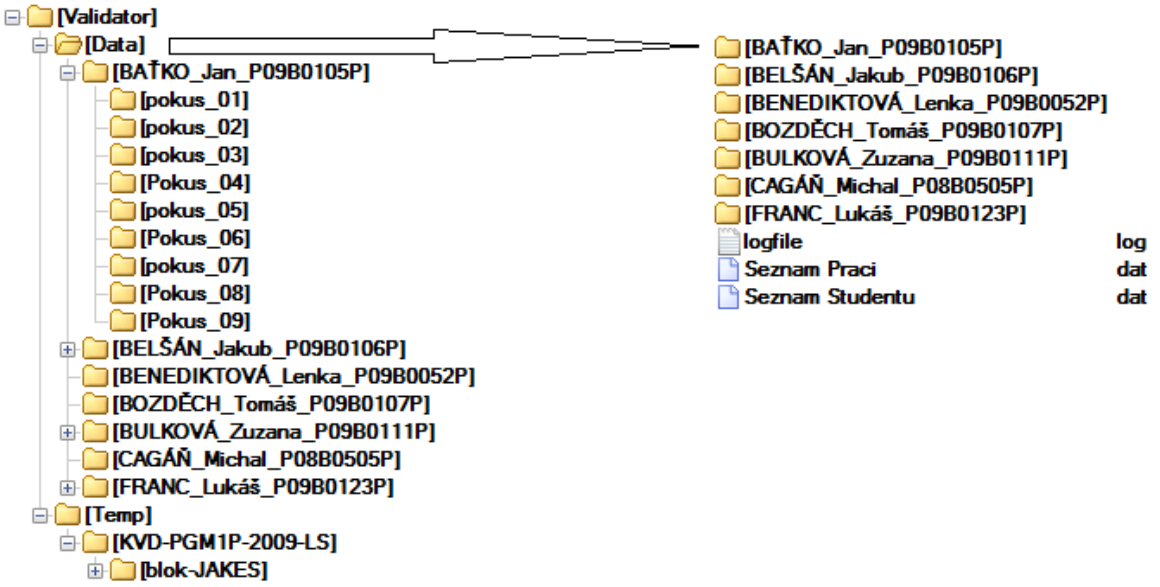
Nesplněno.

Opravu zašlete do sekce speciální.

Obrázek 13 - Ukázka vygenerovaného Microsoft Word dokumentu

8.3 STRUKTURA DAT NA PEVNÉM DISKU

Program pro svou funkci potřebuje ukládat data. Tato data se ukládají do dvou adresářů *Data* a *Temp*. Oba adresáře program vytvoří sám v kořenovém adresáři aplikace. Adresář *Data* slouží k ukládání nově načtených prací do adresářů studentů a zároveň jsou v něm uloženy binární soubory *Seznam Studentu.dat* a *Seznam praci.dat*, které obsahují databáze s údaji, a textový soubor *logfile.log* obsahující záznamy chybových hlášení. Struktura adresářů je zobrazena na obrázku níže – adresář *Validator* je kořenovým adresářem aplikace, v levé části je stromová struktura adresářů a vpravo je detail obsahu adresáře *Data*.



Obrázek 14 - Adresářová struktura programu

9 POPIS ZPRACOVÁNÍ DAT A DŮLEŽITÝCH METOD VE ZDROJOVÉM KÓDU

Základem programu jsou 3 proměnné typu Record, které slouží k ukládání údajů o studentech a načítaných pracích. Jedná se o proměnné TStudent, TPraceCSV a TPrace.

TStudent obsahuje seznam proměnných, které se využívají k identifikaci studenta v programu (jméno, příjmení, studijní číslo atd.). TPraceCSV je datový typ sloužící pro načtení informací z doprovodného CSV souboru při importu prací ze ZIP souboru, tato proměnná je nutná kvůli automatickému načtení data odevzdání, které není možno získat jinou cestou, než načtením z doprovodného CSV souboru. TPrace je poslední proměnná typu Record použitá v programu a obsahuje informace o importované práci. Mezi uložené informace patří např. unikátní ID práce, studijní číslo studenta, který práci vytvořil, zkratka zadání, datum odevzdání a opravy, cesty k jednotlivým souborům .EXE, .PAS a .DOC (případně .DOCX) a příznaky stavu opravy.

Výše zmíněné proměnné se využívají pro vytvoření dynamických polí, která jsou programem zpracovávána. Data do těchto polí jsou importována na programové úrovni. Není nutno všechny záznamy ručně vytvářet a kopírovat například datum odevzdání z CSV souboru ručně. Program obstará automaticky většinu rutinních činností a to během zlomku času, který byl potřeba při manuální evidenci prací. Uživatel poté pouze importovaná data edituje a spravuje pomocí navrženého prostředí programu.

Dynamická pole jsou při manipulaci s daty (změna, nový záznam, vymazání záznamu) ukládána na pevný disk ve formě binárních souborů. Všechny soubory se ukládají do adresářové struktury, kterou si vytvoří program ve svém kořenovém adresáři.

9.1 METODY PRO SPRÁVU STUDENTŮ

Tyto metody slouží k obsluze databáze studentů a nepotřebují složité podpůrné funkce díky jednoduchému způsobu záznamu.

1. **procedure nactiStudenty(sender: tobject);** - tato procedura načte do paměti a zobrazí v komponentě String Grid aktuální seznam studentů
2. **Procedure zapisStudenty(poleStudentuProZapis: array of TStudent);** - slouží pro zápis pole se záznamy studentů do souboru, čímž umožňuje provádět změny nad databází

9.2 METODY PRO SPRÁVU PRACÍ

1. **function importPraciZCSV(souborPraceCSV: string; indexStudijniCislo, indexNazevSouboru, indexCisloPokusu, indexDatumOdevzdani: integer): boolean;**
    a. funkce načte informace z doprovodného CSV souboru do samostatného pole, které slouží pro zjištění data odevzdání práce studentem na webové prostředí Portál ZČU
    b. prvním parametrem je cesta k souboru a dalšími parametry jsou indexy požadovaných informací v CSV souboru
2. **procedure importPraciZeZIP(souborPraceZip: string);**
    a. tato metoda rekurzivně prochází .ZIP soubor s pracemi studentů a vytváří jednotlivé záznamy do databáze
    b. pro správnou funkci je nutno, aby tato metoda měla přístup k informacím z doprovodného CSV souboru, takže je volána až po provedení předchozí funkce
3. **procedure zobrazPrace(seznamPraciProZobrazeni: array of TPrace);** - zobrazí předané pole prací do komponenty String Grid
4. **procedure nactiVsechnyPrace(sender: tobject);**
   procedure nactiPraceByStudijniCislo(studijniCislo: String);
    a. uvedené procedury slouží k načtení všech prací z datového souboru a k zúžení výběru dle studijního čísla
    b. první procedura načítá všechny práce do globální proměnné, která je přístupná celému programu, kdežto druhá z tohoto seznamu vybírá pouze záznamy odpovídající vstupnímu parametru (studijní číslo)
    c. tyto dvě procedury umožňují filtrování a práce s databází prací
5. **procedure zapisPrace(polePraciProZapis: Array of TPrace);** - uloží vstupní pole do datového souboru seznamu prací a tím zajišťuje přidávání a editaci prací v databázi

10 ZÁVĚR

Cílem této práce bylo analyzovat předmět Programování 1 v bakalářském studijním programu Informatika se zaměřením na vzdělávání, prozkoumat možnosti automatické validace studentských prací a navrhnout program, který tuto opravu usnadní.

Po detailní analýze současného postupu opravy studentských prací byla potvrzena skutečnost, že oprava studentských prací je časově velmi náročnou činností. V průběhu roku jsou odevzdávány desítky prací, které je nutné zkontrolovat a zaevidovat.

Každá oprava se skládá z několika opakujících se činností, které jsou spojeny právě s evidencí. Tyto činnosti je možné částečně automatizovat a tím vyučujícímu ušetřit čas strávený opravou.

Navržený a realizovaný program se skládá ze 4 základních částí, které spolu navzájem komunikují a předávají si potřebné informace. Velká část rutinní a opakující se práce byla automatizována a tím i mnohonásobně urychlena. Vytvoření databáze studentů probíhá plně na programové úrovni s možností manuální editace. Import nově odevzdaných prací do databáze je automatizován a zásah uživatele je nutný pouze v případě nedodržení standardů pojmenování či formátu odevzdání studentem. Při opravě prací není nutno procházet jednotlivé složky s příklady na pevném disku, protože program tyto cesty při importu automaticky eviduje a umožňuje k nim jednoduchý a rychlý přístup.

Práce na projektu byla velmi zajímavá. Program využívá několik relativně jednoduchých algoritmů, které společně dohromady dávají funkční celek. Při opravě prací dochází k požadované úspoře času díky automatizaci evidence nových prací a zrychlení přístupu k hodnoceným částem.

11 SEZNAM OBRÁZKŮ

12 SEZNAM LITERATURY

1. **Sobota, L., a další.** *1001 tipů a triků pro Delphi.* Praha : Computer Press, 2002. ISBN 80-7226-529-6.

2. **Keogh, James a Giannini, Mario.** *OOP bez předchozích znalostí.* Brno : Computer Press, 2006. ISBN 80-251-0973-9.

3. **Písek, S.** *Delphi : Začínáme programovat : podrobný průvodce začínajícího uživatele.* Praha : Grada Publishing a.s., 2004. ISBN 80-247-0547-8.

13 RESUMÉ

This thesis analyses seminary works of students of the subject Programming 1 in bachelor's study programme Information science in education at the Department of Computer Science and Educational Technology. It deals with huge amount of works being sent throughout the year to the teacher that demand a lot of time to evaluate and keep a record of.

The thesis contains a deep analysis of the present status from the teacher's point of view. Namely the structure of the subject and conditions required for successfully passing the credit requirement. Furthermore, there is a detailed description of the students' works and present state of evaluating works including all the necessary and fundamental steps taken when evaluating a new work. Also there is a description of commonly made mistakes by students and evaluation standards.

Concluding this work are chapters containing the description and realisation of possibilities of automatic or semi-automatic evaluation of the students' works and basic methodology of detecting cheating that might occur when dealing with the subject's various source codes and documentation that are easy to copy since they are in an electronic form. Lastly there is a technical documentation for a realised computer program that should help teachers with the evaluation of students' works.

14 PŘÍLOHY

1x CD s programem a zdrojovými kódy