Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra Softwarového inženýrství

Bakalářská práce

Nástroj pro výuku programovacího jazyka C a C++

Kateřina Švecová

Vedoucí práce: Ing. Viktor Černý

11. května 2015

Poděkování

Tímto bych chtěla poděkovat svému vedoucímu práce Ing. Viktoru Černému za průběžné konzultace a připomínky. Dále bych chtěla poděkovat své rodině za finanční i psychickou podporu během studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 11. května 2015 .....

České vysoké učení technické v Praze
Fakulta informačních technologií

Odkaz na tuto práci

Švecová, Kateřina. *Nástroj pro výuku programovacího jazyka C a C++*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

Abstrakt

Cílem této bakalářské práce je návrh a realizace interaktivního nástroje pro výuku programovacího jazyka C/C++. Tento nástroj je určen především studentům středních škol a samoukům. V práci najdete důvody, proč jsou současné materiály nedostačující a jaké řešení tohoto problému jsem zvolila. Výsledkem práce je editor pro tvorbu dat a nástroj pro zobrazení těchto dat. Oba nástroje jsou graficky přívětivé.

Klíčová slova Programovací jazyk C a C++, výuka, editor, nástroj, interaktivnost, názornost, GUI, wxWidgets.

Abstract

Aim of my bachelor work is design and realization of interactive educational tool for programming languages C/C++. This tool is dedicated to highschool students and autodidacts. In this work you will find reasons, why contemporary materials are insufficient and what solution of this problem I chose. The result of the work is an editor for data creation and a tool for data display. Both tools are grafically friendly.

Keywords Programming language C and C++, education, editor, visual, tool, interactivity, GUI, wxWidgets.

Obsah

Seznam obrázků

Úvod

V současné době je mnoho studentů a samouků, kteří se chtějí seznámit s programováním a naučit se používat některý programovací jazyk. Bohužel, veliké množství z nich své snažení dříve či později vzdá. Vede je k tomu hned několik důvodů. V loňském roce jsem se setkala se dvěma případy. V tom prvním došlo k tomu, že měli studenti gymnázia ve škole málo ukázek kódu. Navíc byly všechny rozsáhlé a složité na pochopení, což vedlo k celkovému nepochopení látky, ani na základní úrovni. V druhém případě se chtěl student naučit programovat sám. Zakoupil si tedy knihu, podle které se chtěl učit. Tato kniha byla sice kvalitní, ale pro někoho, kdo se nikdy dříve nesetkal s žádným programovacím jazykem, byla složitá a látka v ní se probírala rychle.

Tito studenti si většinou shánějí doučování, v horším případě placené, což jim akorát učení znechutí. Pokud jde o samouky, často své snažení úplně vzdají.

Proto jsem se rozhodla vytvořit interaktivní výukový nástroj, který vysvětlí základy programovacího jazyka C/C++. Zaměřila jsem se především na jednoduchost a srozumitelnost nástroje.

Pro tento účel vznikl také editor, který slouží pro tvorbu uživatelských dat. Tento editor slouží k základním úpravám textu, dále je možnost vkládání obrázků, symbolů, ohraničeného textu. Pro větší přehlednost lze vytvářet systém kapitol a podkapitol, které jsou názorně zobrazeny vedle textu. Tento editor lze samozřejmě použít pro tvorbu jiného obsahu, nejen pro programování.

Jelikož je v dnešní době spousta programovacích jazyků, zaměřila jsem se pouze na jeden, kterým je jazyk C a jeho odvozená objektově orientovaná verze C++. Tento programovací jazyk jsem vybrala především z těchto důvodů. Tím prvním je popularita v dnešní době. Dalším důvodem je časté používání pro výuku na středních a vysokých školách. Posledním důvodem je, že se jedná o multiplatformní programovací jazyk.

Kapitola 1

Cíl práce

Cíl práce

Cílem mé práce je vytvoření interaktivního nástroje pro výuku programovacího jazyka C/C++, který pomůže začátečníkům pochopit základy programovacích jazyků a naučí je vytvořit krátké funkční programy. Tento nástroj bude doplněn o editor, ve kterém se budou vytvářet data. Oba nástroje budou jednoduché, srozumitelné a s dostatečně přehledným grafickým rozhraním.

Editor bude umět, mimo jiné, následující úpravy. Vkládání obrázků. Upravování textu - více typů barev, velikostí a fontů.

Náplň kapitol

První kapitola se zabývá analýzou a návrhem. V části *Programovací jazyky* viz. 2.1 se nejprve krátce zmíním o programovacích jazycích, jejich rozdělení a konkrétně o jazycích C a C++. Následuje diskuze o vývojových prostředích. Z této části vyplývá výběr programovacího jazyka pro výuku a dále volba vývojového prostředí pro ukázku příkladů.

V druhé části 2.2 stejné kapitoly se budu zabývat existujícími řešeními, jejich výhodami a nevýhodami. Tento rozbor vychází z cílové skupiny, kterou jsou studenti středních škol a samouci, kteří se s programováním setkávají poprvé. Z této části vyplývá nový návrh a množina funkcí, které by měl program splňovat.

Následuje kapitola *Realizace* 3, ve které najdete použité knihovny a přidané funkce do programu. V sekci *Problematické části* 3.3 se zmíním o konkrétních problémech, které nastaly při implementaci, případně jejich řešení (pokud jsem toto řešení našla). Tato část by měla posloužit budoucím čtenářům, kteří by narazili na stejné problémy.

V další částech 3.4 je uveden systém řešení, jak ukládat data. V závěru kapitoly se zabývám samotnými výukovými daty a použitými styly.

Poslední kapitola 4 je o testování výsledné implementace, nalezených chybách a jejich řešení.

Na konci práce najdete závěr, ve kterém jsou shrnuty výsledky práce a výhled do budoucna, který skýtá návrhy na rozšíření programu.

Kapitola 2

Analýza a návrh

2.1 Programovací jazyky

Následující kapitoly jsou teoretické. Celá kapitola se zaobírá existujícími informacemi, ze kterých jsem, za pomoci níže uvedených zdrojů, vytvořila stručné informace o programovacích jazycích. Čerpala jsem z následujících zdrojů: *Učíme se programovat v jazyce* C od V. Kadlece, *Učebnice jazyka C* od P. Herouta a *Programování v C++* od M. Viriuse. Dále jsem použila tyto internetové zdroje: stránka o programovacích jazycích s názvem *PROGRAMOVACÍ JAZYKY* dostupné z [4] a článek *Programovací jazyk* ze stránek české Wikipedie, dostupné na [5].

2.1.1 Obecně

Co jsou programovací jazyky a jak je můžeme rozdělit?

Programovací jazyk je nástroj, pro zápis programu. Podobně jako český jazyk, má svá pravidla, která se musí dodržovat. Každý programovací jazyk má svá klíčová slova, která se používají k jeho zápisu. Programovací jazyk je tedy způsob komunikace mezi programátorem a počítačem. Ke komunikaci potřebujete znát jednak slova jazyka, ale také pravidla, jak tyto slova používat. Je to velmi podobné jako u českého jazyka. Avšak programovací jazyky jsou přísnější a pokud jím daná pravidla nedodržíte, nebude Vám vůbec rozumět. Oproti cizím jazykům, kde si můžete dovolit drobné chyby například v gramatice, bude Vám okolí i tak rozumět a dále s Vámi komunikovat.

Programovacích jazyků existuje veliké množství. Můžeme je také rozdělit do několika skupin, podle různých kritérií. Níže jsou uvedena tři základní kritéria dělení.

Programovací jazyk je tedy velmi podobný klasickým jazykům, kterými se dorozumíváme. Je zde však jeden veliký rozdíl. Pokud mluvíte ať už česky, anglicky, nebo německy, stačí Vám naučit se pouze příslušná slova, pravidla, fráze. Pokud vše ovládnete, bez problému se dorozumíte. Chcete-li však pro-

gramovat, nestačí jen říci počítači, co po něm chcete, aby udělal. To by bylo moc snadné a programátorem by mohl být každý. Musíte mu říci **JAK**, ještě lépe mu zdělit veškeré možnosti a podrobnosti. Na jednu stranu se potřebujete naučit samotný programovací jazyk (ta lehčí část), ale zbytek musíte vymyslet sami.

Vyšší a nižší

Nižší programovací jazyky jsou velmi primitivní. Jde téměř o strojové instrukce procesoru, programátor tedy musí i ty nejjednodušší příkazy zapsat podrobně a do detailů.

Vyšší programovací jazyky mají větší míru abstrakce. Programátor se tak může zaměřit na samotnou myšlenku programu (logickou část), jsou lépe srozumitelné a pracuje se v nich rychleji a efektivněji. Tyto jazyky bývají přenositelné. Mezi tyto jazyky patří například: *C, C++, Java, Python, PHP, Delphi, Perl*.

Interpretované a kompilované

Kompilované jazyky se musejí před prvním spuštěním vždy zkompilovat. Mezi tyto jazyky patří mimo jiné *Pascal*, *C* a *Java*.

Oproti tomu interpretované jazyky jsou překládány až za běhu programu. Proto jsou také pomalejší, ale na druhou stranu nemají tak složité požadavky na správně zapsaný kód (nedeklarují se proměnné, nepoužívají se ukazatele). Nejznámějším jazykem je *Python*, dále *Ruby* a *Perl*.

Procedurální a neprocedurální

Procedurální, neboli imperativní jazyky. Popisuje se jak. Jde o posloupnost příkazů, které určují celý postup řešení. Odborně se tento postup označuje jako algoritmus.

Zatímco u neprocedurálních (deklarativních) jazyků se řekne, *co* se má dělat, ne jak.

2.1.2 Jazyk C

Zde uvedu základní charakteristiky jazyka C. V současné době je C jeden z nejpoužívanějších jazyků. Jedná se o snadno pochopitelný jazyk, naučit se základy není nic složitého. Další výhodou je multiplatformnost - v tomto jazyce lze programovat na různých platformách, stejně tak výsledné programy lze vytvořit funkční pro různé platformy. Pokud již umíte jazyk C, můžete snadněji přejít na jeho objektovou verzi C++. Lze vytvořit kvalitní a efektivní programy. Jazyk C je dobrým základem pro programátora, dobře se navazuje v učení nových jazyků. Například jazyk *Python* je také velmi zajímavý, dá se říci snadnější než-li jazyk C. Programátor ani nemusí deklarovat proměnné,

jsou zde datové typy, které nejsou typické pro ostatní jazyky. Tyto dvě usnadnění jsou jak výhodou, tak nevýhodou. Pokud se programátor začne učit na tomto jazyce, přijde mu pak zvláštní, proč by se vůbec s psaním datových typů obtěžoval apod. Z tohoto důvodu se *Python* nehodí pro začátky. Stejně jako ostatní interpretované jazyky.

Právě z výše uvedených důvodů se jazyk C velmi často používá pro výuku ve školách. Další alternativou na školách je jazyk *Pascal*. Tento jazyk byl vlastně vytvořen jako jazyk pro výuku programování, ale v dnešní době se pro psaní programů a projektů v podstatě nepoužívá a výuka *Pascalu* přechází ve výuku *C*.

2.1.3 Jazyk C++

Programovací jazyk C++ je objektové rozšíření jazyka C. I přesto zde při programování narazíte na konstrukce pocházející z jazyka C, které nelze použít v jazyce C++. Další funkce lze použít u obou jazyků, ale pokaždé mají jiný význam. Takových případů je naštěstí málo, ale mohou způsobit potíže.

2.1.4 IDE Code::Blocks

Jelikož jsem pro výuku vybrala jazyk C a C++, potřebuji zvolit vývojové prostředí. Toto prostředí bude doporučené uživatelům na tvorbu programů. Také se zde budou pořizovat snímky kódů, které budou doprovodným materiálem k textu.

V dnešní době se nabízí mnoho vývojových prostředí (dále IDE), ale je nutné si vybrat jedno a v tom provést všechny ukázky. Tato diskuze se tedy týká výběru IDE pro jednotlivé snímky a stejně tak bude toto IDE doporučeno uživatelům, pro jejich práce. Tento výběr přímo neovlivňuje výběr IDE, ve kterém budu tvořit program já sama.

Vývojové prostředí

Vývojové prostředí se často označuje zkratkou IDE [6]. Tato zkratka pochází z prvních písmen celého anglického názvu, který zní: Integrated Development Environment.

Vývojová prostředí pomáhají programátorovi při práci. Jde o graficky přívětivý software, ve kterém se nachází editor kódu, kompilátor(interpret) a často debugger. Většina IDE jsou navržena pro konkrétní programovací jazyk/y, aby se lépe přizpůsobilo programovacím paradigmatům daného jazyka. Vývojová prostředí mohou být určena konkrétní platformě, ale mohou být stejně tak multiplatformní.

Konkrétní IDE

Zde jsou uvedeny jedny z nejznámějších vývojových prostředí.

- **Microsoft Visual Studio** Velmi rozsáhlý nástroj pro mnoho programovacích jazyků. Velikou nevýhodou je však, že se jedná o produkt firmy Microsoft, tedy jde o placený software. Navíc není multiplatformní.

- **NetBeans** NetBeans jsou multiplatformní a volně dostupné ke stažení. Primárně jsou však určeny programovacímu jazyku Java. V základním balíčku je podpora pro jazyky C i C++.

- **DEV-C++** Vývojové prostředí společnosti Bloodshed software. Je určeno pro jazyky C a C++. Bohužel je určen pouze pro operační systém Windows.

- **Code::Blocks** Toto prostředí je multiplatformní, určené pro jazyky C a C++. Navíc je bezplatné, jelikož je zapsáno pod GNU GPL licencí, jedná se o open source. Je vyvíjeno v C++ a využívá grafickou knihovnu (toolkin) wxWidgets.

Shrnutí

V předchozích sekcích, především v části o jazyku C viz. 2.1.2 jsem uvedla důvody výběru jazyků C/C++ pro výuku prvního programovacího jazyka.

Jako doporučené IDE, pro tvorbu programů, bude doporučeno vývojové prostředí Code::Blocks. Stejně tak v tomto prostředí budou pořízeny všechny snímky, screeny, které budou vloženy do výukových materiálů. Výhodou je především zaměření na C/C++, multiplatformnost a bezplatnost programu. Proto jsem se rozhodla právě pro tento software.

2.2 Existující řešení

V současné době existuje mnoho výukových podkladů pro programovací jazyky. Lze sehnat velmi kvalitní materiály. Hlavním problémem je cílová skupina. Je mnohem jednodušší učit se nový programovací jazyk, když už nějaký znáte. Většina knih a internetových návodů jsou určeny právě těm, kteří již mají alespoň nějaké základy. Právě pro začátečníky je obtížné začít a osvojit si svůj první programovací jazyk. Dalším problémem je názornost, grafické zpracování, počet ukázek. Zde jsou největší problémy. V následujících kapitolách je kladen důraz na to, že se jedná o začátečníky a žáky gymnázií, případně středních škol, kteří se s programováním ještě nikdy nesetkali. Tento fakt je pro celou moji práci stěžejní.

Nejčastěji se samouci učí z knih a internetových stránek. Proto jsem se na ně zaměřila. Pro konkrétní ukázku jsem vybrala dvě knihy a dvě internetové stránky. Tyto materiály se zabývají právě jazyky C/C++.

2.2.1 Kniha

Jednou z nejznámějších českých knih o programování je Učebnice jazyka C [7] od **Pavla Herouta**. Tato kniha vyšla ve dvou částech, první a druhý díl. Veškeré základy najdete v dostatečné podobě již v prvním díle. První díl je určen pro začátečníky, ale i středně pokročilé. Druhý díl je určen alespoň středně pokročilým.

Kniha je členěna do kapitol a podkapitol. Výklad je srozumitelný, jsou uváděny různé případy použití. Pokud již umíte programovat, je to nepochybně skvělá volba, jak se naučit další programovací jazyk.

Nevýhody

Začátečník může mít problémy s přílišnou odborností textu. Dalším problémem je mnoho nových informací a rychlé tempo. Text není nijak zbarven, ani zde nejsou rámečky. Členění vzniká různými fonty, velikostí, případně tučností a kurzívou.

Výhody

Členění do kapitol a podkapitol je velmi přehledné. Velkým kladem jsou upozornění autora. Autor zde uvádí: poznámky, co je dobré si uvědomit a časté chyby. V průběhu kapitol jsou příklady, na konci příklady k řešení.

Virius

Podobně je na tom velmi rozsáhlá učebnice (téměř 500 stránek) s názvem Programování v C++ od Miroslava Viriuse [8]. Oproti knize pana Herouta se tato učebnice zabývá pouze programovacím jazykem C++. Jelikož se jedná o objektovou formu jazyka C, zabývá se autor již od začátku funkcemi, které se v C zpravidla probírají až v druhé části knih, či jiných materiálů. Tato učebnice je pro začátečníky ještě náročnější, než-li výše uvedená.

Na druhou stranu jde o velmi kvalitní knihu. Členění je podobné jako v *Učebnici programování jazyka C*. Najdete zde teorii, příklady, poznámky. Pro pokročilejší bych tuto knihu doporučila, ne však pro úplné začátečníky.

2.2.2 Webové stránky

Výuka programovacího jazyka C bývá na většině stránek podobná. Na těch lepších jsou jednotlivé kapitoly členěny v pravém, či levém sloupci pro rychlou a snadnou orientaci. Samotný výklad bývá jednolitý. Především nebarevný text a členění do rozsáhlých objemných odstavců. Tento text je nepřehledný a po chvíli čtení se v něm člověk akorát ztrácí. To je nepochybně veliká nevýhoda.

Výhody

Téměř vždy je výklad doplněn o ukázky. Buď jako text (barevný, nebo nebarevný), nebo jako snímky pořízené přímo z vývojového prostředí a příslušný výstup. Občas jsou tyto ukázky vymezeny rámečkem. Je tomu právě díky webovým stránkám, kde se autor nemusí omezovat v rozsahu. Na obrázcích (popř. snímcích) je vše vidět dobře, oproti knihám, kde jsou autoři co se týče snímků velice omezeni.

Sally

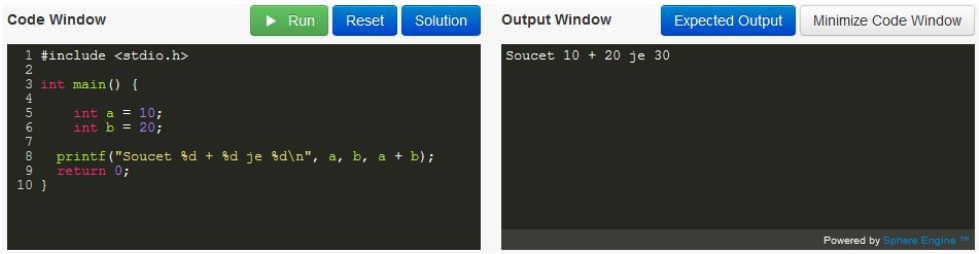
Tato stránka [9] je členěna do dvou části. Vlevo je systém kapitol, uprostřed samotný výklad. Zaujalo mě zde vkládání ukázek a výstupů. Ukázka kódu je vložena do speciálního rámečku, který sám provádí formátování kódu. Tento kód pak vypadá, jako by byl ve vývojovém prostředí. Výstup programu je v rámečku s bílým písmem a černým pozadím (simulace výstupu do konzole).

Learn-C

Tento web [1] mě zaujal možností kompilace přímo na webové stránce, ukázka na obrázku 2.1. O tuto kompilaci se stará Sphere engine [10] a nabízí velkou škálu programovaích jazyků k okamžitému překladu.

Na samotné stránce jsou ukázky kódu, které jsou v rámečcích. Pod každou ukázkou je možnost spuštění. Konkrétní ukázka se nahraje do překladače (levá část na výše zmíněném obrázku), následně se zkompiluje a výstup se zobrazí v pravé části. Stejně tak si můžete do editoru (levé části) vytvořit vlastní kód a spustit si ho (otestovat, zda funguje jak má). Tento materiál je v anglickém jazyce.

Obrázek 2.1: Snímek online kompilátoru[1]



2.2.3 Problémy současného řešení

V předešlé části jsem již poukázala na některé problémy existujících řešení. Zde jsem udělala souhrn těch nejdůležitějších.

Materiály nejsou dostupné v českém jazyce

Pro tuto věkovou skupinu je velmi složité učit se novou látku z cizojazyčných podkladů. Ať už se jedná o knihy, aplikace, či internetové zdroje.

Veliké množství materiálů

Uživatel si v takovém případě není schopen vybrat. Pokud mu někdo přímo nedoporučí konkrétní materiál, zpravidla knihu, jen těžko si vybere nějakou vhodnou právě pro něj. Může si vyhledat v internetových diskuzích, ale právě tam často odpovídají zkušenější uživatelé a ne začátečníci. Najdou si tam skvělé materiály, ale ne pro začátečníky.

Cena

Tištěné materiály i některé dostupné aplikace jsou drahé. Cena bývá velkou demotivací k prvotnímu kroku a odhodlání začít.

Názornost

Některé knihy, popř. internetové zdroje, obsahují veliké množství textu, který však není nijak speciálně zabarven, zvýrazněn a obohacen o snímky kódu. Především grafika a názornost jsou pro studenty v tomto věku důležité.

Málo ukázek

Většinou jsou texty doplněny o ukázky, ale těchto ukázek je málo a nepokrývají různé použití dané problematiky. Někdy jsou uvedeny lehké příklady, jindy pouze složité (velice komplexní). Potřeba je ale uvést oboje.

Málo příkladů k řešení

Často tyto příklady k řešení úplně chybí. Avšak jsou velice důležité. Navedou uživatele vyzkoušet si psát kód. Při tom si uživatel zjistí, co má za úskalí daná problematika a lépe si ji osvojí.

Chybí ukázky řešení

Bohužel se stává, že autoři doplnili výukové materiály o příklady k řešení, ale nenabídli žádné ukázky řešení k zadané úloze. Zde jsou dva základní problémy. Uživatel nedokázal přijít na řešení a tudíž nezná vůbec žádné řešení. Nebo uživatel napsal jiné řešení, což je samozřejmě naprosto v pořádku, ale může tak přijít o řešení z jiného úhlu pohledu a to mu do budoucna může chybět.

2.2.4 Podobné materiály

V této kapitole se budu věnovat materiálům, které se sice nezabývají přímo výukou programovacího jazyka C/C++, ale jsou dobře zpracovány a proto je beru jako inspiraci pro vlastní projekt.

Zvon

Prvním zdrojem jsou internetové stránky, které se, mimo jiné, zabývají *XML, XPathem, XSLT*. Tyto stránky jsme používali jako podporu v předmětu Technologie XML. Zde je ukázka, jak vypadá část stránky výukových podkladů.

Na obrázku 2.5 patrné rozdělení do kapitol (vpravo), při kliknutí se načte stránka s aktuální problematikou. Lze se tedy jednoduše, rychle a efektivně přesouvat mezi kapitolami. Na začátku stránky je vždy krátký úvod do tématu a dále pokračují ukázky použití. Ukázek je většinou více (jak vidíte, členění pomocí rámečků), aby pokryly různé případy. V jiné sekci si může uživatel sám vyzkoušet zadané příklady.

Ookami

Další zajímavou stránkou je výukový materiál pro programování v jazyce *Python*. Tato stránka obsahuje vždy odkaz na dané téma, které je dále rozděleno do kratších úseků. Jednotlivé stránky se přepínají šipkami nebo kliknutím na příslušné tlačítko.

V ukázkách níže jsou tři po sobě následující stránky (úseky). Na první, obrázek 2.2, jsou uvedeny různé případy použití. Na druhé straně, obrázek 2.3, jsou ukázky použití s příslušnými výstupy. Třetí strana, obrázek 2.4, obsahuje příklady k řešení. Řešení těchto příkladů se nenachází hned na další straně, ale je více odděleno. Právě tento systém mi přijde velice efektivní.

- Uvedení do tématu (obecně, teoreticky), rozbor různých použití.
- Ukázky použití (různá, jednoduchá, názorná), výstupy kódů.
- Příklady k řešení (cílené na procvičení více typů použití), oddělené ukázky řešení (nesmí chybět).

Z těchto materiálů jsem se sama učila. Učilo se mi efektivně a když jsem něco nevěděla, rychle jsem na těchto stránkách našla pomoc.

Zajímavým prvek je také obarvení kódu pro větší přehlednost. Názvy datových typů, příkazů a funkcí jsou barevně odděleny od zbytku kódu.

Důležité jsou i poznámky, které jsou psané méně výraznou šedou barvou, aby byly odděleny od zbytku textu. Nacházejí se v nich poznámky, upozornění a různé další dodatky k textu.

Obrázek 2.2: Snímek internetové stránky - teorie[2]

Cyklus „for“

I. Narozdíl od snad naprosté většiny programovacích jazyků se v Python'u cyklí průchodem po prvcích nějakého objektu, ve většině případů sekvence:

```
for i in SEKVENCE:
   BLOK
```

→ Navíc narozdíl od většiny ostatních jazyků je proměnná cyklu (zde `i`) definována na stejné úrovni kódu jako sama smyčka, takže po skončení cyklu bude stále existovat a mít právě tu hodnotu, jíž během provádění cyklu dosáhla (zde poslední prvek procházené sekvence).

II. Přímou obdobou klasického *for*-cyklu je pak průchod po prvcích odpovídajícího rozsahu:

```
for i in range(INTERVAL):
   BLOK
```

III. Naopak zcela neklasickou je varianta s větví *else*:

```
for i in SEKVENCE:
   BLOK 1
else:
   BLOK 2
```

Větev s `else` se provede pouze tehdy, je-li cyklus ukončen normálně, tedy dojde-li k proiterování všech prvků vstupní sekvence. Naopak je-li cyklus v *BLOKu 1* ukončen násilně pomocí `break`, kód v *BLOKu 2* nebude vůbec vvkonán.

Obrázek 2.3: Snímek internetové stránky - příklady použití[2]

Příklady – „for“

I. Zde je vstupním objektem, po jehož prvcích se iteruje, seznam tří řetězců:

```
>>> for word in ["welcome", "to", "python"]:
...     print(word)
...
welcome
to
python
```

II. Po tomtéž seznamu zde iterujeme za dodatečné pomoci funkce `enumerate()`, která navíc vrací pořadí prvků v objektu:

```
>>> for i, word in enumerate(["welcome", "to", "python"]):
...     print(i, word)
...
0 welcome
1 to
2 python
```

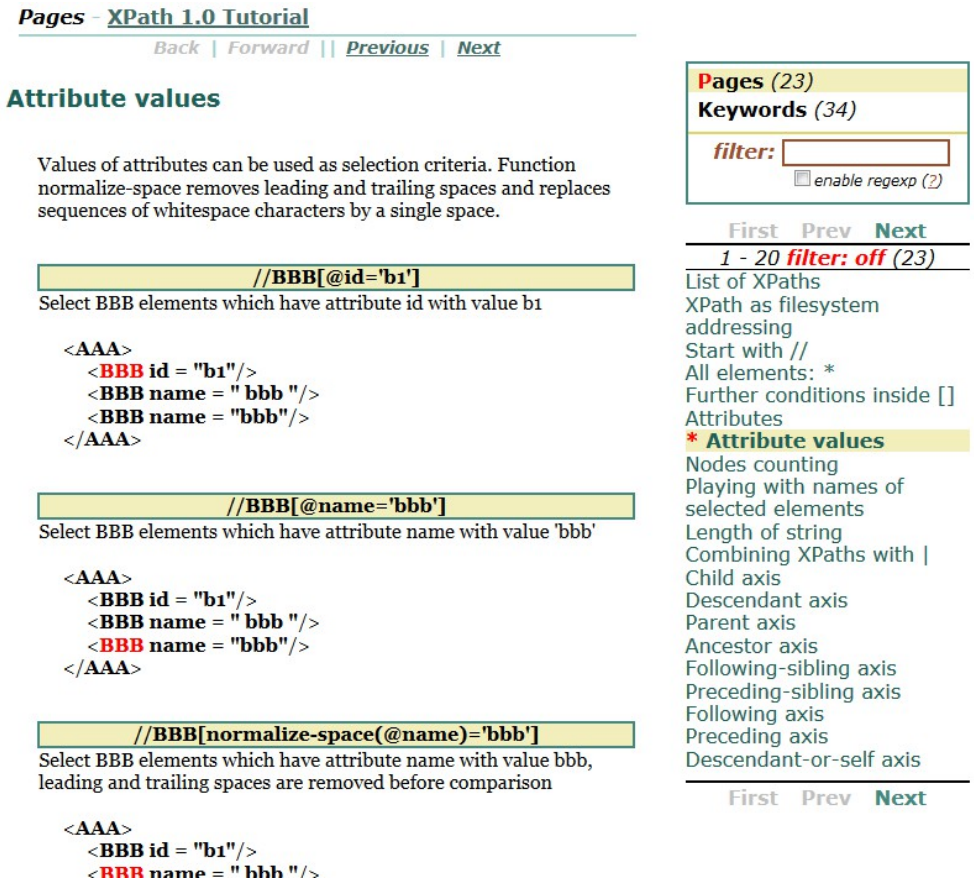
→ Všimněte si použití n-tice na místě řídících proměnných cyklu.

Obrázek 2.4: Snímek internetové stránky - příklady k řešení[2]

Cvičení

1. Vypište si postupně všechna písmena ve vašem jméně.
2. Vypište všechna čísla od 0 do 10.
3. Vypište 11x pozdrav *Ahoj!*
4. Sečtěte čísla mezi 0 a 10.
5. Rozšiřte předchozí příklad – spočítejte průměrnou hodnotu z daných čísel. (Použijte funkci *len()* na příslušný rozsah.)
6. Vypište z vašeho jména každý druhý znak.
7. Vypište z vašeho jména každý třetí znak.

Obrázek 2.5: Snímek rozvržení výukového materiálu[3]



2.3 Návrh funkcí

Zde jsou základní požadované funkce na rozhraní. Tyto funkce jsou všechny vyžadovány. Mohou se samozřejmě rozšířit a obohatit o další. Následuje pouze nutné minimum funkcí, aby mohl program fungovat a splňoval zadání. Následující funkce se týkají editoru, zobrazovač bude obsahovat podmnožinu těchto funkcí.

Funkce na správu souborů

Program bude mít základní funkce pro práci se soubory. V adresáři budou dvě složky. Jedna složka pro editované projekty (složka *projects*), druhá pro projekty ke zobrazení (složka *exports*). Komunikace mezi složkami je znázorněna na obrázku 2.7. Systém načtení dat je zobrazen na diagramu viz. 2.8.

- **Nový projekt** - vytvoří se nový projekt, který bude mít pouze jednu stránku, tedy i jednu kapitolu. Uživatel si zvolí název projektu. Aktuálně otevřený projekt se zavře. Pokud bude projekt s tímto názvem již existovat, program nahlásí chybu.
- **Otevřít projekt** - otevře se již existující projekt. Při otevření bude zobrazena nabídka dostupných projektů (ze složky *projects*).
- **Uložit projekt jako** - vytvoří se kopie aktuálního projektu. Uživatel bude dotazován na název nového projektu. Starý projekt se zavře, otevře se pod novým zvoleným názvem stejný právě vzniklý obsah.
- **Uložit aktuální stránku** - uloží se aktuálně načtená stránka. Pokud si uživatel zvolí akci, při níž se změní, nebo zavře aktuálně zobrazená neuložená stránka, bude upozorněn a dotázán na její uložení.
- **Exportovat** - Aktuálně zobrazený projekt se exportuje do složky projektů ke zobrazení (*exports*).
- **Zavřít** - Celý projekt se uzavře. Pokud není aktuální stránka uložena, je uživatel dotázán.

Funkce na úpravu textu

Editor nabídne základní úpravy textu. V diagramu 2.6 je komplexní použití funkcí na editaci textu. Jsou zde i funkce, které byly do editoru přidány navíc (označeny žlutě), více v kapitole 3.2.

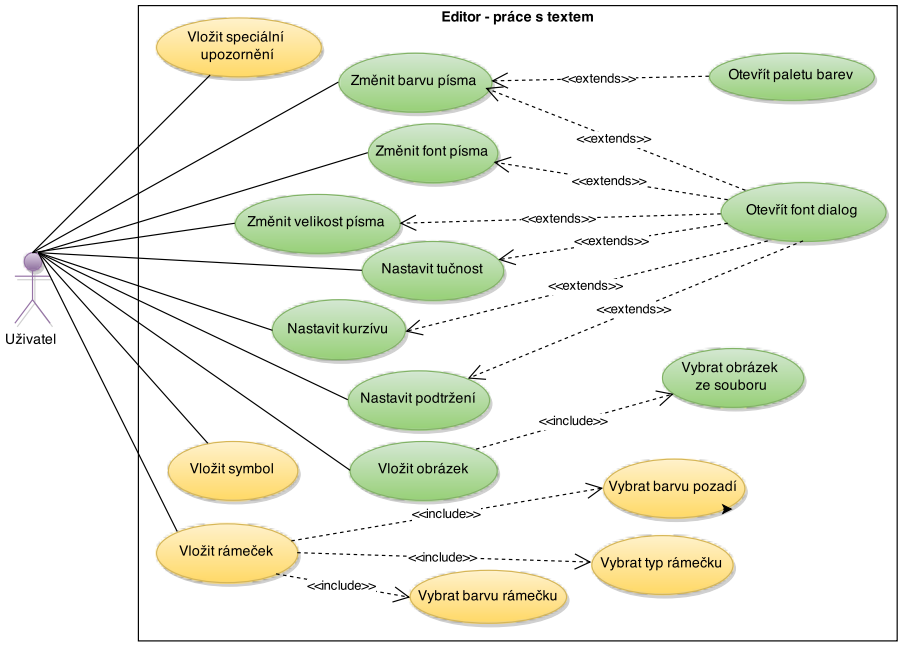
- **Velikost** - volba z předem definovaných velikostí.
- **Font** - nabídka dostupných fontů.
- **Barva písma** - výběr z palety barev.

- **Zarovnání** - nabídka zarovnání vlevo, vpravo a na střed.
- **Vkládání obrázků** - vkládání obrázku různých formátů s možností nastavené velikosti při vkládání.

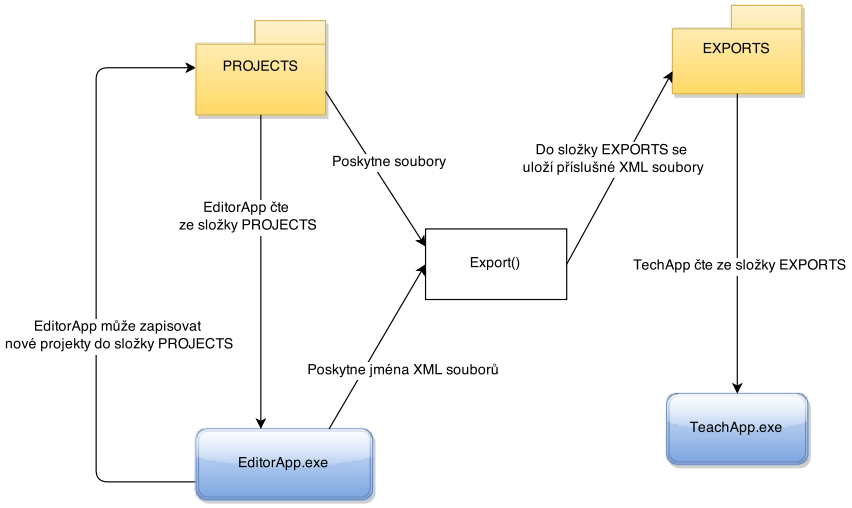
Funkce pro systém kapitol

- **Vložení kapitoly** - vloží se nová kapitola stejné úrovně, jako je kapitola, na které bylo vyvoláno menu. Tato kapitola se vloží nakonec. Uživatel bude dotázán na jméno kapitoly.
- **Vložení podkapitoly** - podobně jako vložení kapitoly, ale vloží se podkapitola k aktuálně zvolené kapitole.
- **Posun o jedno výše/níže** - slouží pro posouvání (záměnu) kapitol na stejné úrovni. Tedy pokud bude vytvořena struktura: první kapitola, druhá kapitola, třetí kapitola a třetí kapitolu posuneme o jedno výše, bude nová struktura: první kapitola, třetí kapitola, druhá kapitola.
- **Přejmenovat** - přejmenování kapitoly.
- **Smazat** - smazání kapitoly i s celým systémem podkapitol mazané kapitoly. Uživatel bude upozorněn.

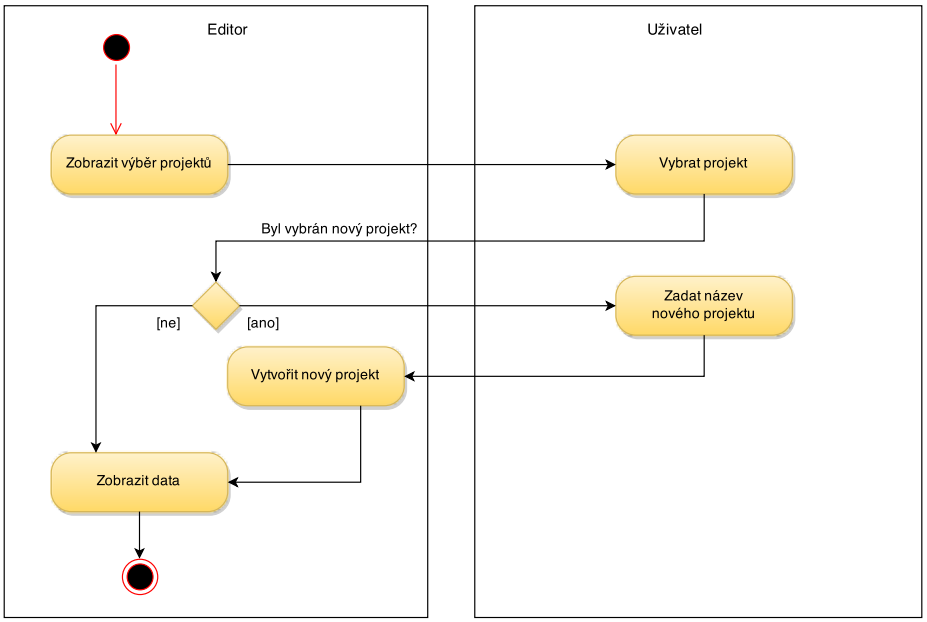
Obrázek 2.6: Diagram případů užití pro editaci textu



Obrázek 2.7: Diagram znázorňující komunikaci mezi složkami a programy



Obrázek 2.8: Diagram znázorňující načtení dat do editoru



2.4 Výběr programovacího jazyka

Následující kapitola pojednává o výběru programovacího jazyka pro vlastní implementaci editoru a zobrazovače. Vybrala jsem si programovací jazyky C/C++. Vedly mě k tomu následující výhody.

- **Multiplatformní** - tyto jazyky jsou multiplatformní (mohou být spuštěny na různých platformách), není tedy potřeba se výběrem programovacího jazyka omezit na konkrétní platformu. Jelikož bude nástroj určen těm, kteří se s programováním nikdy nesetkali, pravděpodobně budou používat některou z verzí Windows, ale pro další rozšíření a obohacení programu je zbytečné zahodit multiplatformní použití jenom z tohoto důvodu. Výuková data budou také v multiplatformním jazyce, proto je třeba mít nástroj, který budou moci uživatelé použít na různých OS.

- **Stejný jazyk jako náplň** - program poslouží jako ukázka toho, co se dá v programovacím jazyku C/C++ napsat.

- **Grafická knihovna wxWidgets** - při analýze kódu jiného programu jsem se seznámila s touto knihovnou, kterou lze použít, mimo jiné, právě pro jazyky C/C++. Během analýzy jsem narazila na komponenty, které by se daly využít pro mou bakalářskou práci.

  Jelikož jsem nikdy s žádnou externí grafickou knihovnou nepracovala, chtěla jsem se to naučit. Navíc tato knihovna nabízela všechny potřebné třídy a funkce. A jako bonus byla použita pro tvorbu IDE Code::Blocks.

2.5 Návrh GUI

Nové řešení vychází z výše diskutovaných kladů a záporů současných řešení. Velký důraz je přitom kladen na cílovou skupinu, kterou jsou studenti středních škol a samouci, kteří se s programování nikdy nesetkali.

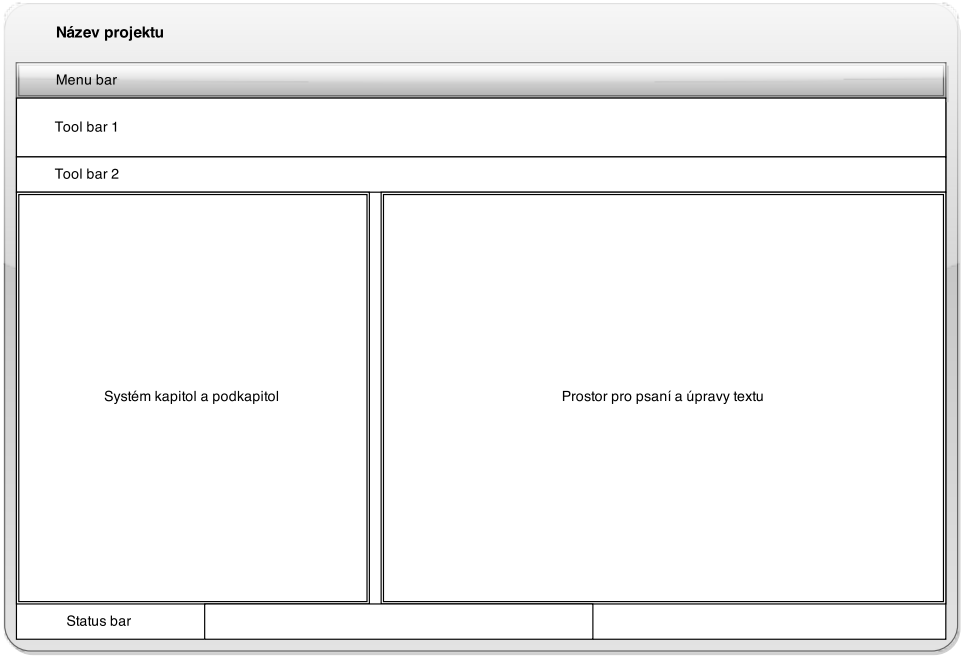
2.5.1 Návrh editoru

Výukové prostředí bude mít podobu klasického okna. To bude obsahovat následující komponenty 2.9. Pro tvorbu jsem použila webovou aplikaci *Draw.io* [11].

- **Menu bar** (panel menu) - menu bude horizontální. Jednotlivé položky se budou rozbalovat vertikálně. Pod menu barem se budou nacházet dva horizontální toolbary.

- **Toolbar 1** (panel nástrojů) - první toolbar (odshora) bude sloužit pro větší tlačítka. Plánovaná velikost je 32x32 pixelů.

Obrázek 2.9: Návrh GUI - rozložení komponent



- **Toolbar 2** - v dolní liště budou menší tlačítka, velikost 16x16 pixelů. Tento druhý panel bude pouze u editoru, jelikož se zde budou nacházet tlačítka pro úpravu textu, která nebudou u zobrazovače potřeba.
- **Status bar** - tato lišta je horizontální a nachází se v dolní části okna. Zpravidla se zde zobrazují nápovědy, aktuální verze a některé podrobnosti.
- **Systém kapitol** - v levé části se bude nacházet systém kapitol a podkapitol v podobě stromové struktury. Aktuálně zobrazená stránka bude zvýrazněna, odlišena viditelně od ostatních (zde tučně).
- **Prostor pro text** - místo, kam bude uživatel psát svůj vlastní text, měnit ho a vkládat obrázky. U zobrazovače se bude text pouze zobrazovat, nebude již možno do textu zasahovat.

2.5.2 Návrh zobrazovače

Jelikož již nelze ve zobrazovači provádět změny, je zde pouze minimum funkcí na manipulaci programem.

- **Otevřít projekt** - otevře existující projekt ze složky *exports*.
- **Zavřít** - zavře projekt.

- **Tisk** - tisk aktuálně zobrazené stránky.
- **Náhled tisku** - náhled tisku.

Kapitola 3

Realizace

3.1 Knihovny

Pro oba programy jsem použila grafickou knihovnu wxWidgets [12]. Žádnou další knihovnu jsem nepoužívala, ani nebyla třeba. WxWidgets má vlastní třídy pro řetězce, práci se soubory apod. Tuto knihovnu jsem vybrala ze tří důvodů.

- **Open source** - knihova je volně dostupná.
- **Mutliplatformní** - jedná se o multiplatformní knihovnu.
- **Vlastní důvody** - v předmětu Softwarový týmový projekt jsme měli za úkol zanalyzovat hudební aplikaci *Audacity*. Domovská stránka viz. [13]. Já jsem dostala za úkol provést analýzu použité grafické knihovny wxWidgets. Do této doby jsem s žádnou grafickou knihovnou pro jazyk `C/C++` nepracovala. Díky této analýze jsem již dříve zjistila, které funkce/třídy jsou v této knihovně dostupné. Pro tuto aplikaci jsem pouze zkontrolovala, jestli existují funkce pro stromovou strukturu kapitol a možnost vytvářet text se základními úpravami. V neposlední řadě se dá tato knihovna použít i pro jazyk Python, který jsem se také učila. Proto jsem se rozhodla vybrat knihovnu wxWidgets, jelikož splňuje požadavky na moji aplikaci a zároveň se ji chci naučit používat.

3.1.1 Použité třídy

Zde jsou uvedeny ty nejdůležitější třídy z wxWidgets, které jsou nezbytně zapotřebí. Úplným základem je vytvoření nové aplikace, která dědí od *wxApp*. Pro koncept okna je použita třída *wxFrame*.

Hlavní třídy

Zde jsou uvedeny třídy, které odpovídají základním komponentám z obrázku 2.9. *WxMenuBar*, *wxMenu*, *wxToolBar*, *wxStatusBar*, *wxTreeCtrl*, *wxRichTextCtrl*.

Podpůrné třídy

Tyto třídy jsou nezbytné k tomu, aby byly hlavní komponenty správě umístěny a také se dalo manipulovat s poměrnou velikostí mezi šířkou systému kapitol a šířkou textu. Jedná se o třídy: *wxSplitterWindow*, *wxPanel*, *wxBoxSizer*.

Ostatní

Tyto třídy jsou potřebné k fungování jednotlivých komponent a zbytku programu. Pro práci s řetězci je tu *wxString*. Dále jenom výčet tříd: *wxTreeItemId*, *wxRichTextXMLHandler*, *wxRichTextBuffer*, *wxBitmap*, *wxComboBox*, *wxMessageDialog*. Ostatní třídy nejsou tolik důležité, slouží k bližší specifikaci jednotlivých částí.

3.2 Přidané funkce do editoru

Pro větší přehlednost a usnadnění jsem přidala další funkce, které nebyly přímo určeny v zadání. Jde o následující funkce.

- **Tisk** - tisk aktuálně zobrazené stránky.
- **Náhled tisku** - náhled tisku.
- **Předdefinované barvy** - pro častou a rychlou změnu barvy nabízí editor 13 tlačítek s barvami. Kliknutím na barvu se označený text obarví stejnou barvou, jako je zobrazena na tlačítku. V případě pouhého kliknutí do text (žádné označení) se barva nastaví a nově napsaný text na tomto místě bude nést zvolenou barvu.
- **Poslední uložená barva** - tlačítko, které si pamatuje poslední použitou barvu. Pokud si uživatel vytvoří vlastní barvu v paletě barev, může si ji přidat do vlastních barev. Při každém znovu použití by musel znovu zavolat celý dialog s paletou. Právě pro časté používání uživatelem vytvořené barvy je zde tlačítko, které si ji za něj zapamatuje.
- **Vložení symbolu** - občas je třeba vložit speciální symbol. Tento symbol lze vytvořit klávesovými zkratkami, nebo konkrétním speciálním fontem. Pro rychlejší a snadnější vkládání jsem přidala tlačítko pro vkládání symbolů.

- **Vložení rámečku** - Pro větší přehlednost vkládaných ukázek kódu jsem také přidala možnost vložit rámeček. Uživatel si může pro větší přehlednost zvolit barvu pozadí rámečku, typ ohraničení (plně, čárkovaně, tečkovaně) a barvu rámečku.

- **Vložení speciálního obrázku** - přímo v programu je možnost vložit obrázek typu *JPG*, velikosti 48x48 pixelů. Editor nabízí 24 těchto obrázků.

- **Vložení speciálního upozornění** - vloží se symbol a následně rámeček. Jsou zde tři druhy, pro upozornění, informaci a otázku.

- **Přidání tlačítek se šipkami** - dvě nová tlačítka, šipkou vlevo se posunete o kapitolu výše, tedy na předchozí kapitolu. Šipkou vpravo níže, na následující kapitolu. Přesuny fungují pouze na stejné úrovni pro stejného rodiče (a samozřejmě pokud to je možné).

3.3 Problematické části

Při implementaci jsem narazila na některé funkce, se kterými jsem měla problémy. Pročítala jsem internetové diskuze a dokumentace wxWidgets a snažila jsem se je vyřešit. Ve většině případů na tyto problémy, nebo velmi podobné situace, narazili již jiní uživatelé. Řešení těchto problémů mi zabralo zbytečně mnoho času, proto zde uvedu ty nejdůležitější poznatky, aby mohly pomoci případně někomu dalšímu.

Malé doporučení, pokud voláte funkci, které musíte dát jako parametr některý event, aby se vůbec zavolala, ale tento event nepoužijete, budete mít při kompilaci upozornění. Těch se jednoduše zbavíte, když obalíte název eventu makrem `WXUNUSED(*)` viz. 3.1.

Kód 3.1: Ukázka kódu funkce pro zachycení místa vyvolání popupmenu

```
OnTreeExpand(wxCommandEvent& WXUNUSED(event))
```

Posouvání kapitol

Třída *wxTreeCtrl* má mnoho funkcí. Například lze jednoduše sbalit, či rozbalit všechny kapitoly. Proto jsem také tuto funkci přidala, jelikož by se uživateli mohla hodit pro rychlejší zorientování, čeho všeho se materiál týká. Při mazání, přejmenování a vkládání nebyl nijak velký problém. Ten nastal, když jsem se snažila vytvořit funkci pro prohazování kapitol. Četla jsem mnoho diskuzí, ale tato funkce v příslušné třídě není. Všechny diskuze vedly pouze k jediné volbě viz. [14]. Je třeba vybrat jednu větev a celou ji znovu rekurzivně vybudovat nad (případně pod, záleží na volbě a konkrétním řešení) tou

druhou. To znamená vytvořit stejnou strukturu se stejně pojmenovanými potomky a zachovat jejich vlastnosti.

Pro větší přehlednost je lepší také rozbalit cestu, na které se nachází zobrazená stránka, pokud se jí týká prohazování, aby se uživateli neztratila z dohledu.

Popupmenu

V tomto případě se jedná spíše o drobnost, která však může působit velké problémy, pokud si neuvědomíte, proč se tak děje. Při kliknutí pravým tlačítkem na *tree control* a následné zvolení volby, se ztratí informace o tom, na které položce bylo toto menu vyvoláno. Je třeba si zapamatovat místo, odkud se následně vyvolá *popupmenu*. Já jsem si zvolila proměnou lastTreeClick, která je deklarována v private a jedná se o *wxTreeItemId*. Konkrétní příklad viz. 3.2.

Kód 3.2: Ukázka kódu funkce pro zachycení místa vyvolání popupmenu

```
void MyFrame::OnRightClickTree(wxTreeEvent& event)
{
        lastTreeClick = event.GetItem().GetID();
        PopupMenu(popupMenu);
}
```

Aktuální volba písma

Při kliknutí do textu je třeba zobrazit, jak je text právě formátován. Zobrazí se aktuální font, velikost, druh zarovnání a případně tučnost, kurzíva nebo podtržení.

Nejprve jsem zkoušela reagovat na event, který se vyvolá při kliknutí do rich textu - `wxEVT_RICHTEXT_LEFT_CLICK`. Ten fungoval správně. Pokud tedy uživatel používal pouze myš, bylo by vše v pořádku. Problém nastal při použití šipek. Při pohybu šipkami na jiný text s jiným nastavením. Ve wxWidgets je také event na zachycení zmáčknutí klávesy. S tímto eventem (`EVT_KEY_DOWN`) bývá problém, alespoň podle mnoha diskuzí a reportů, které jsem našla na internetu. Ten se vůbec nespustil, nereagoval na žádné stlačení klávesy. V diskuzích se psalo o nastavení stylu `wxWANTS_CHARS` rodičovi, které většinou pomohlo. Mně se však event stále nespouštěl, ani pro použití stylu.

Nakonec jsem našla v ukázkách použití (*samples*) wxRichTextu eventy `EVT_UPDATE_UI`. Tento event se stále volá a lze tak kontrolovat aktuální nastavení. Například pro tučnost se při zavolání eventu stačí podívat, zda-li je aktuální text tučný, či není. Tím, že se zavolá funkce `void Check (bool check)`. Té řeknete, jestli je tučné (`true`), nebo není (`false`). Tato funkce se pak sama postará o zaškrtnutí možností v menu, zmáčknutí/vymáčknutí ikonky apod. Samozřejmě je potřeba mít tento event svázán stejným ID jako menu, popř. toolbutton.

Kopírování složek

Při zavolání funkce *Uložit projekt jako*, je třeba překopírovat celou složku se všemi soubory. Ve wxWidgets neexistuje funkce, která by překopírovala celou složku (multiplatformně) i se všemi soubory. Dokonce zde není funkce pro přímé zkopírování souboru z jedné složky do druhé. Řešení je následující: vytvoříte si systém složek a podsložek se stejnými názvy, zatím prázdné. Procházíte postupně veškeré soubory, případně rekurzivně složky. Každý soubor otevřete, načtete a uložíte do nového souboru na správné místo. V tomto případě jsem načetla soubor jako stream, který jsem přímo přesměrovala do výstupního streamu. V následující ukázce viz. 3.3 jsou vynechány některé, pro tento konkrétní problém, nedůležité řádky. Jedná se o kontrolu, zda jsou streamy v pořádku, případné vyhození chybového hlášení. Nejprve si vytáhneme jméno prvního souboru ve složce, pokud existuje vrací funkce `true`, v opačném případě `false`. Následně si bereme další názvy, dokud funkce vrací `true`.

Kód 3.3: Ukázka kódu pro kopírování složky

```
wxDir::Make(nameOfFolderProjects
  + textEntryDialog->GetValue());
wxDir::Make(nameOfFolderProjects
  + textEntryDialog->GetValue() + "/XML");

wxDir dir(nameOfProject + "/XML");
wxString name;

bool next = dir.GetFirst(&name);

while (next)
{
  wxFileInputStream inputStream(nameOfProject +
  "/XML/" + name);

  wxFileOutputStream outStream(nameOfFolderProjects
  + textEntryDialog->GetValue() + "/XML/" + name);

  inputStream.Read(outStream);
  next = dir.GetNext(&name);
}
```

Blikající kurzor ve zobrazovacím módu

Při zobrazení textu je nežádoucí, aby se v textu zobrazoval blikající kurzor (po kliknutí levým tlačítkem myši do textu). Nastavit rich text tak, aby se do něj nedalo psát a měnit jej je snadné, stačí použít funkci `SetEditable(false)`

s parametrem `false`. Tato funkce ale nezabrání, aby se nezobrazoval kurzor po kliknutí do textu. Jedná se sice vizuálně o detail, ale zároveň jde o rušící element. Nejprve jsem se snažila reagovat na event, který zachytával kliknutí levým tlačítkem myši do textu a následně tento focus přesměrovat jinam. Toto řešení sice fungovalo, ale na pohled to vypadalo divně. Nakonec jsem opět hledala v diskuzích, kde jsem také po odladění vytvořila funkční řešení viz. 3.4.

Kód 3.4: Ukázka kódu pro kopírování složky

```
richTextCtrl->Bind(wxEVT_SET_FOCUS,
                   &MyFrame::OnMouse, this);

void MyFrame::OnMouse(wxFocusEvent& WXUNUSED(evt))
{
        richTextCtrl->GetCaret()->Hide();
}
```

Tisk

Poté co jsem přidala možnosti na tisk, jsem zkoušela tisknout různá nastavení textu, ale také obrázky a rámečky. Tisknutí fungovalo, ale na vytištěné stránce byly dvě věci odlišné, než-li v programu. Přitom náhled tisku ukazoval vše stejně, jako tomu bylo v programu, ale tisk se mírně lišil.

Tisk obrázků s alfa kanálem

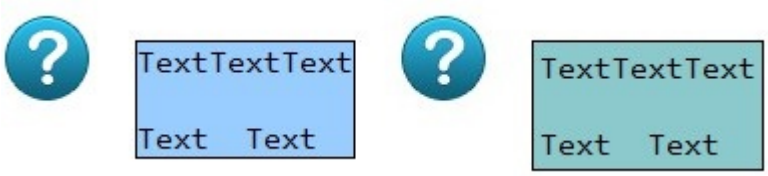
Pokud si vkládáte obrázky do textu a obsahují alfa kanál, tedy jsou na některých místech průhledné, je vše v pořádku zobrazeno. Na místě průhlednosti je vidět barva pozadí. Problém nastává při tisku. V náhledu je vše jako v editoru, ale při tisku se alfa kanál převede na černou barvu. Snažila jsem se nastavit, aby se alfa kanál změnil na barvu pozadí. Byla zde jedna funkce, která však nefungovala korektně. Řešení jsem nenašla. Ale jelikož jsou oba programy určeny především ke zobrazování, ne k tištění, dále jsem se tímto problémem nezabývala. Jediné, co jsem předělala, byly předdefinované obrázky. Ty se sice na tlačítku zobrazují jako *PNG*, ale vkládají se jako *JPG*. To také přináší jedno omezení. Pokud si vložíte tento obrázek do rámečku s barevným pozadím, bude se okolí obrázku zobrazovat bíle. Jelikož nepředpokládám použití těchto obrázků pro vkládání do rámečků, nechala jsem je jako *JPG*. Na obrázku 3.1 vlevo, pokud vložíte *PNG*, vpravo *JPG*. Při vkládání vlastních obrázků zůstane problém s černým alfa kanálem při tisku.

Obrázek 3.1: Ukázka png a jpg formátu

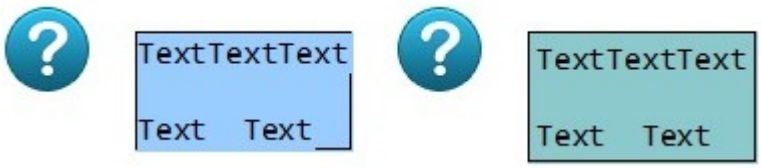
Tisk rámečků

Při tisku rámečků nebyl rámeček souvislý (samozřejmě pro případ volby klasického nepřerušovaného stylu). Byl na několika místech narušen. Stalo se tak, protože zde končil text, nebo šlo o dolní část, kde byl text. Na obrázku 3.2 je vidět snímek přímo z editoru, na druhém obrázku 3.3 je snímek z náhledu tisku (stejně narušeně se i rámeček vytiskne). Nalevo je zastaralý problematický způsob řešení, vpravo nový funkčně správný.

Obrázek 3.2: Ukázka rámečků v editoru



Obrázek 3.3: Ukázka rámečku v náhledu pro tisk



Řešení je velmi snadné, stačí přidat malé vnitřní odsazení rámečku a text již dále nebude narušovat strukturu rámečku. Kód viz. 3.5

Kód 3.5: Nastavení vnitřního odsazení textu

```
attr.GetTextBoxAttr().GetTopPadding().SetValue(3,
        wxTEXT_ATTR_UNITS_PIXELS);
attr.GetTextBoxAttr().GetBottomPadding().SetValue(3,
        wxTEXT_ATTR_UNITS_PIXELS);
attr.GetTextBoxAttr().GetLeftPadding().SetValue(3,
        wxTEXT_ATTR_UNITS_PIXELS);
attr.GetTextBoxAttr().GetRightPadding().SetValue(3,
        wxTEXT_ATTR_UNITS_PIXELS);
```

3.4 Ukládání jednotlivých stránek

Jednotlivé projekty jsou uloženy ve stejnojmenné složce. Každá stránka (kapitola) má uložený svůj vlastní XML soubor. Tento soubor je pojmenován podle všech rodičů kapitoly, doplněné svým vlastním názvem. Proto jsou také přidána některá opatření. Při vytváření kapitol stejné úrovně se stejným rodičem je kontrolován nový název při vytváření, či přejmenování. Pokud se tento název shoduje s některým sourozencem kapitoly, vyhodí se upozornění. Další kontrolou prochází samotný název, jelikož nesmí soubory obsahovat v názvu některé znaky. Při zobrazování nové stránky se musí soubor nahrát do bufferu. Díky tomuto systému pojmenování souborů se konkrétní jméno ke zvolené stránce vytváří efektivně. Pouze se dotazuji rekurzivně rodiče na jméno.

Pro systém kapitol je v projektu vytvořen samostatný soubor, ve kterém se nachází na každém řádku dva údaje. Prvním je celočíselná hodnota zanoření, druhým je název kapitoly. Pokud je zanoření rovno -1, nastaví se stránka uvedená o jedno výše jako aktuální a označená při otevření souboru. Toto zobrazení naposledy otevřený stránky je vhodné především při práci v editoru. Uživatel se tak vrátí k poslední stránce, kterou měl zobrazenou.

Práce s XML

V dokumentaci knihovny wxWidgets jsem našla funkce, které mi usnadnily práci s načítáním a ukládáním XML souborů. Pro načítání a ukládání dat jsem použila třídy *wxRichTextBuffer* a *wxRichTextXMLHandler*. V ukázce kódu viz. 3.6 jsou nejdůležitější funkce. Pro načítání a ukládání XML souborů se používá buffer, který získáte přímo z *wxRichTextCtrl* a stream (vstupní/výstupní). O samotné parsování XML souborů se postará sama knihovna wxWidgets. Jedinou nevýhodou je, že Vám dovolí *rich text control* použít pouze jeho vlastní objekty a styly, ale to v mojí práci nevadilo, jelikož jsem zde našla všechny potřebné funkce.

Kód 3.6: Ukázka kódu pro práci s XML soubory

```
wxFileInputStream inputStream(name);
//...
xmlHandler->LoadFile(&richTextCtrl->GetBuffer(),
        inputStream);
richTextCtrl->Refresh();

wxFileOutputStream outputStream(name);
//...
*outputBuffer = richTextCtrl->GetBuffer();
xmlHandler->SaveFile(outputBuffer, outputStream);
```

3.4.1 Export

Co vlastně dělá funkce export s daty? Abych mohla odpovědět na tuto otázku, musím zde nejprve uvést, jak vypadá samotná složka se soubory. Následuje struktura 3.4, kde jsou uvedeny pouze nezbytné části pro pochopení práce se soubory.

Obrázek 3.4: Struktura souborů

```
exports ..... adresář se všemi exportovanými projekty
projects ..... adresář se všemi projekty k editování
EditorApp.exe ..... spustitelná forma editoru
TeachApp.exe ..... spustitelná forma zobrazovače
```

Ve složce *projects*, se nachází jednotlivé složky projektů, které jsou pojmenovány stejnojmenným názvem, jako je jméno projektu. Stejně tak je tomu u složky *exports*, kde se nacházejí všechny exportované projekty. Pokud chcete pracovat v editoru, spustíte *EditorApp.exe*, pro spuštění zobrazovače *TeachApp.exe*. V obou případech budete dotázání na název projektu. Dostanete nabídku jmen projektů. Tato nabídka se generuje podle názvů projektů v dané složce (u editoru je navíc možnost nového projektu). Podle výběru se otevře příslušný projekt.

Dále je třeba vědět, co přesně se děje při mazání kapitol a přidávání podkapitol. Vše vychází z výše zmíněného systému vytváření XML souborů, ke každé stránce kapitoly. Pokud se z nějaké kapitoly, která dříve neměla žádné podkapitoly, stane rodič a bude mít podkapitolu, nepůjde již dále zobrazit její obsah. Následuje příklad viz. 3.5 a 3.6.

Obrázek 3.5: Původní hierarchie kapitol

```
Kapitola1
    Kaptiola11
Kapitola2
Kapitola3
```

Obrázek 3.6: Nová hierarchie kapitol

```
Kapitola1
    Kaptiola11
Kapitola2
Kapitola3
    Kapitola31
```

Jak vidíte, do původní hierarchie kapitol viz. 3.5, byla přidána nová kapitola se jménem *Kapitola31* viz. 3.6. Díky tomu se z kapitoly *Kapitola3* stal

rodič a proto již nelze zobrazit její obsah. Proto by se také mohl smazat její obsah, jelikož již dále nebude potřeba. Podobně je tomu u funkce *Smazat*.

Nakonec jsem se rozhodla, že tyto soubory mazat nebudu. A proč? Každému se může stát, že si vytvoří podkapitolu a následně mu dojde, že jí tam nechce a chtěl by se vrátit k původnímu textu. Ten by byl však nenávratně smazán. Při mazání tedy mažu pouze hierarchii, ale samotné soubory smazány nejsou. Proto zde také existuje funkce *Export*. Ta se zeptá na jméno exportované složky, do té se následně přesunou pouze soubory, které jsou v chvíli zobrazitelné. V exportovaných složkách tedy nejsou žádné zbytečné soubory navíc. Pokud si někdo vytvoří výukový materiál a vyexportuje si ho, může tento export dále poskytnout, jelikož v něm již nejsou přebytečné soubory.

Ještě malý detail. Pokud je mezi mazanými i kapitola, která je aktuálně zobrazena (je mazána přímo tato kapitola, nebo je mazán její rodič), najde se první možná kapitola, která se zobrazí místo ní. Hledání začíná v první kapitole a jde rekurzivně na její potomky, dokud nenarazí na kapitolu, která již dále nemá podkapitoly (je listem).

3.5 Výuková data

3.5.1 Použité styly

Data jsem přehledně rozdělovala do kapitol a podkapitol, přesně jak bylo zamýšleno. V textu jsem použila různé velikosti a barvy písma, případně tučnost a kurzívu (možnost podtržení jsem nepoužívala). Fonty jsem používala zpravidla dva, jeden pro text, druhý pro kód. Samotné ukázky kódu byly vymezeny v rámečcích a proměnné, datové typy a další důležité prvky byly odděleny barvou, případně ztučněny.

Dále jsem do textu vkládala obrázky, které byly pořízeny jako snímky přímo z vývojového prostředí a jejich případné výstupy na konzoli. Tyto snímky jsem používala, aby pomohly začátečníkům co nejvíce se zorientoval v novém prostředí (vývojovém). Proto jsem použila snímky pouze ze začátku, dále pouze ukázky kódu jako text v rámečku.

Ke každé kapitole, pokud to bylo možné, jsem vytvářela tři stejné podkapitoly. Byly to následující tři:

- **Teorie** - úvod do problematiky. Vysvětlení různých použití, doplněno o komentáře, případné upozornění a doporučení.
- **Ukázky/příklady** - zadání příkladu, ukázka(y) řešení. Rady, typy, triky.
- **Příklady k řešení** - zadání úkolu k řešení. Případně doplnění o rady, upozornění. Nenachází se zde řešení příkladů, aby nesvádělo uživatele se podívat dříve, než se pořádně zamyslí a sám to zkusí. Přeci jenom

chybami se člověk učí. Samotné řešení se nachází ve speciální kapitole, kde jsou oddělena do podkapitol řešení z jednotlivých okruhů.

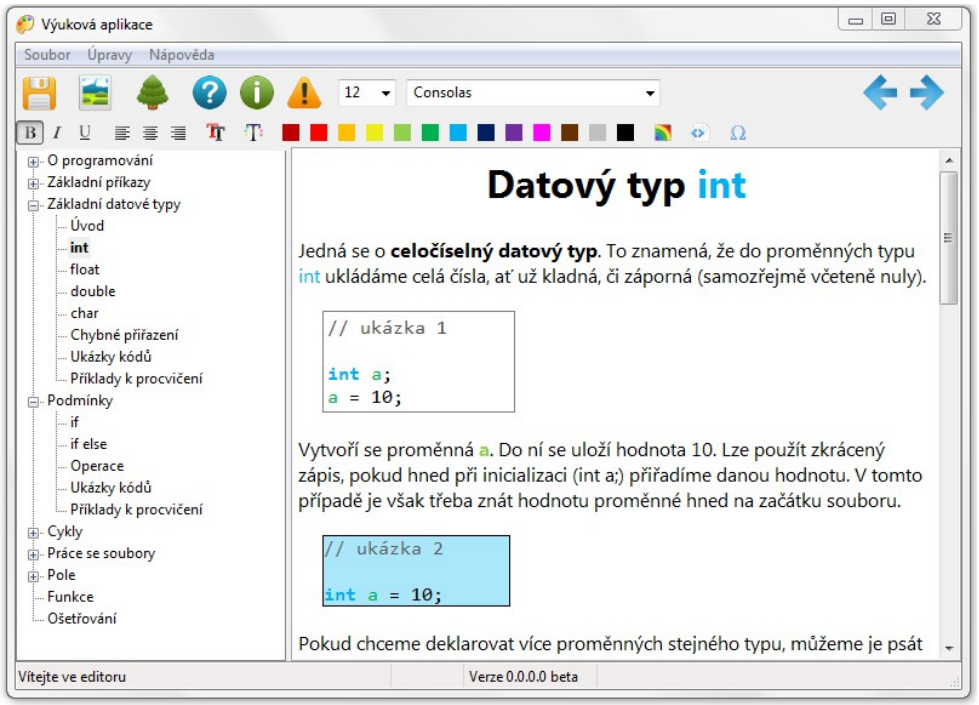
3.5.2 Výběr témat

Jako ukázku, jak by mohly vypadat data na výuku jsem zvolila pouze jazyk C. Objektovou verzi jsem zatím pro náplň daty nepoužila. Jde opravdu o ukázku, jelikož mým oborem studia není učitelství, ani se nezabývá teoretickými částmi programování. Hlavním cílem mé práce je návrh a realizace výukového nástroje s jeho funkcemi, které jsou vymezeny požadavky na rozhraní.

3.5.2.1 Konkrétní použití

Zde je ukázka 3.7 výsledného prostředí. Návrh hlavních komponent byl dodržen. Všechna zobrazená tlačítka jsou funkční.

Obrázek 3.7: Ukázka hotového editoru



Úplně nahoře na liště je uvedeno jméno projektu (tak, jak si uživatel projekt pojmenuje). Pod ním následuje klasické menu, dále dvě nástrojové lišty. V levé části je systém kapitol, vpravo je část pro editaci textu.

Kapitola 4

Testování

Testování jsem prováděla já sama, jelikož znám program a vím, kde jsou problematické úseky. V testování mi dále pomáhala rodina (pohled uživatel, kterému je tento program určen a nezná nic o vývoji). Při testování byly objeveny některé chyby, více v následujících kapitolách. Příslušný kód reportovaných chyb byl opraven.

4.1 Testování editoru

Při testování editoru byly objeveny následující chyby.

Označení aktuálně používané stránky v systému kapitol

Při dvojkliku pravým tlačítkem myši na kapitolu se buď tato kapitola zobrazí (pouze v případě, že tato kapitola nemá žádné další podkapitoly), nebo se všechny podkapitoly sbalí/rozbalí (opačný krok k aktuálně použitému).

Při používání šipek k přechodu mezi kapitolami stejné úrovně, tedy sourozenci jsem nastavila, že příslušná tlačítka zšednou, tedy nepůjdou použít. Toto omezení fungovalo správně.

Přesto nastala chyba, jelikož jsem nekontrolovala, zda-li je příslušný sourozenec rodičem, nebo není. Pokud nebyl a neměl již žádné potomky, nebyl problém. Chyba nastala v případě, že se přesouvalo na kapitolu mající další podkapitoly. Tato kapitola totiž nemůže být zobrazena jako text.

Smazání všech stránek

Pokud se mazala kapitola, byly smazány i všechny její podkapitoly (pokud existovaly). Při zavolání příkazu na smazání všech kapitol (smazání se vyvolalo na jediného potomka kořene), se program zasekl a následně proběhl pád programu.

Vytváření souboru s nepovolenými znaky

Názvy souborů mají svá pravidla, nesmí být použity některé znaky. Uživatel byl dotazován na obyčejný text, který nebyl podroben žádné kontrole a přímo z tohoto textu byl vytvářen soubor/složka.

Byla vyhozena chyba přímo od programu. Ten na to nebyl nijak připraven a data neseděla jak mají, proto se již dále ukazovaly chyby vyvolané touto původní chybou.

Chybné zobrazení názvu projektu

Při otevření projektu se jeho název objeví vlevo nahoře jako název programu. Při otevření se název zobrazoval správně. Pokud došlo k výběru možnosti v menu *Otevřít projekt*, nebo *Nový projekt*, nezobrazil se korektně správný název, zůstal tam starý. U možnosti *Uložit projekt jako* se nový název zobrazil správně.

4.1.1 Oprava chyb

Všechny chyby byly opraveny. Samotná oprava chyb nebyla problematická.

4.2 Testování zobrazovače

V tomto případě byla objevena pouze jedna chyba, která se shodovala s výše uvedenou. Šlo o kontrolu posouvání šipkami.

Závěr

Výsledek práce

Výsledkem práce je editor pro vytváření dat a zobrazovací program, kde se data pouze zobrazují, ale nelze je již měnit.

Editor umožňuje vytvoření systému kapitol a podkapitol. K jednotlivým kapitolám, které již nemají žádně další podkapitoly, se zobrazuje příslušný text. Tento text je možno editovat. Jedná se o úpravy textu (zarovnání, velikost, barva a font písma, tučnost, kurzíva a podtržení), dále vkládání obrázku (různé formáty), symbolů a textu do rámečku. Tento editor je navržen tak, aby byl přívětivý a jednoduchý na ovládání pro každého uživatele. V editoru lze také tisknout aktuálně zobrazenou stránku, což nebylo původně zadáno, ale ukázalo se jako vhodné.

Zobrazovací program načte data vytvořená editorem a zobrazí je v konkrétním formátu. V tomto módu již nelze data měnit, pouze prohlížet a tisknout.

Původním záměrem bylo vytvořit nástroj pro výuku programovacího jazyka C/C++. Tento nástroj vznikl tak, jak byl vymezen. Navíc lze použít editor a zobrazovač pro různá data. Vhodné využití je například pro dějepis či zeměpis. Díky systému kapitol a podkapitol lze vytvořit přehledný výukový materiál. Vhodné je pak využít vkládání obrázků a úpravu textu. Nakonec lze některé stránky vytisknout (dějepis - výpis dat a událostí, zeměpis - výpis států a příslušných měst, vodstva, pohoří apod.).

Vznikl tak ucelený nástroj pro tvorbu a zobrazení uživatelského obsahu. V editoru byla vytvořena data pro výuku programovacího jazyka C, které lze zobrazit. Tedy cíl mé bakalářské práce byl splněn.

Výhled do budoucna

První možností je pokračovat v původní myšlence a rozšířit nástroj o vlastní kompilátor. Zde je však složité zachovat multiplatformnost, jelikož se kompilátory chovají rozdílně na jednotlivých platformách. Dalším rozšířením v pro-

gramovací části je doplněk programu na editaci kódu. Tím je myšleno, že by obarvení proměnných, případně dalších částí kódu probíhalo automaticky. Jazyky C a C++ mají svá vlastní klíčová slova. Uživatel by si vybral označení těchto slov (například modře a tučně). Také by šlo přidat vlastní kategorie a do nich vlastní slova (výrazy). Tyto slova by se zobrazovaly všechny podle konkrétního určení nastavení písma kategorie.

Druhou možností je rozšíření editoru o další funkce. Ukázalo se, že dá nástroj využít pro další výuku, nejen programovacích jazyků, ale například biologie, dějepisu, zeměpisu apod. Proto se nabízí rozšíření pro matematiku a fyziku. V této chvíli zde není možnost, jak efektivně vkládat vzorečky, rovnice a další matematické výpočty (maximálně jako snímky). Studenti, nebo i učitelé by pak mohli vysvětlit danou látku, zadávat úlohy, poskytovat řešené úlohy apod. právě díky těmto dvěma nástrojům.

Literatura

[1] Variables and Types. [online], © Learn-C.org, [cit. 2015-04-29]. Dostupné z: http://www.learn-c.org/en/Variables_and_Types

[2] Znamenáček, J.: přehled Python'u. [online], [cit. 2015-04-29]. Dostupné z: http://vyuka.ookami.cz/materialy/python/overview.0.xml

[3] Nič, M.: XPath 1.0 Tutorial. [online], [cit. 2015-04-29]. Dostupné z: http://zvon.org/comp/r/tut-XPath_1.html#Pages~Attribute_values

[4] PROGRAMOVACÍ JAZYKY. [online], [cit. 2015-05-06]. Dostupné z: http://k-prog.wz.cz/progjaz/

[5] Programovací jazyk. [online], 2015, [cit. 2015-05-06]. Dostupné z: http://cs.wikipedia.org/wiki/Programovac%C3%AD_jazyk

[6] Vývojové prostředí. [online], 2014, [cit. 2015-05-06]. Dostupné z: http://cs.wikipedia.org/wiki/V%C3%BDvojov%C3%A9_prost%C5%99ed%C3%AD

[7] Herout, P.: *Učebnice jazyka C*. České Budějovice: Kopp, 6 vydání, 2009, ISBN 978-80-7232-383-8.

[8] Virius, M.: *Programování v C++*. V Praze: České vysoké učení technické, třetí vydání, 2009, ISBN 978-80-01-04371-4.

[9] Programování v jazyku C/C++. [online], 2003, [cit. 2015-04-29]. Dostupné z: http://www.sallyx.org/sally/c/

[10] Sphere Engine. [online], © 2014, [cit. 2015-04-29]. Dostupné z: http://sphere-engine.com/

[11] Draw.io Online User Manual. [online], 2014, [cit. 2015-05-06]. Dostupné z: https://support.draw.io/display/DOD/Draw.io+Online+User+Manual

[12] WxWidgets. [online], © 2015. Dostupné z: http://wxwidgets.org/

[13] Audacity. [online], 2015, [cit. 2015-05-07]. Dostupné z: http://web.audacityteam.org/

[14] wxTreeCtrl move/swap/copy? [online], 2012, [cit. 2015-04-30]. Dostupné z: https://forums.wxwidgets.org/viewtopic.php?t=33974

Příloha **A**

Seznam použitých zkratek

GPL General Public License

GUI Graphical User Interface

IDE Integrated Development Environment

OS Operating System

XML eXtensible Markup Language

XPath XML Path Language

XSLT eXtensible Stylesheet Language Transformations

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `exe` ..... adresář se spustitelnou formou implementace
  - `data_images` ..... adresář s obrázky potřebnými pro oba programy
  - `exports` ..... adresář se všemi exportovanými projekty
  - `projects` ..... adresář se všemi projekty k editování
  - `EditorApp.exe` ..... spustitelná forma editoru
  - `TeachApp.exe` ..... spustitelná forma zobrazovače
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `BP_Švecová_Kateřina_2015.pdf` ..... text práce ve formátu PDF