Univerzita Karlova
Pedagogická fakulta

BAKALÁŘSKÁ PRÁCE

2018 Ondřej Kolín

Univerzita Karlova

Pedagogická fakulta

Katedra informačních technologií a technické výchovy

BAKALÁŘSKÁ PRÁCE

Výuka vývoje softwarových aplikací s GUI na střední odborné škole

Application development with GUI at a secondary vocational school

Ondřej Kolín

Vedoucí práce:	PhDr. Jiří Štípek, Ph.D.
Studijní program:	Specializace v pedagogice
Studijní obor:	Informační technologie se zaměřením na vzdělávání

2018

Prohlašuji, že jsem bakalářskou práci na téma „Vývoj aplikací s uživatelským rozhraním v prostředí střední odborné školy“ vypracoval pod vedením vedoucího práce samostatně za použití v práci uvedených pramenů a literatury. Dále prohlašuji, že tato práce nebyla využita k získání jiného nebo stejného titulu.

Praha 20. 04. 2018

.....

podpis

Rád bych poděkoval PhDr. Jiřímu Štípkovi Ph.D. za vedení mé bakalářské práce, za vstřícnost při řešení problémů, rady a pomoc při tvorbě této práce.
Rád bych také poděkoval kolektivu Střední průmyslové školy Mladá Boleslav za milé přijetí do pracovního kolektivu a za podporu při studiích. Jmenovitě bych rád poděkoval Ing. Václavu Bohatovi za důvěru, kterou ve mně vložil přijetím jako učitele odborného výcviku a panu Ing. Martinu Kubátovi za skvělý osobní příklad jakožto pedagoga a nadřízeného.

ANOTACE

Tato bakalářská práce se zaměřuje na srovnání možností výběru programovacích jazyků a technik pro vývoj aplikací s grafickým uživatelským rozhraním. Práce se snaží podchytit alespoň v základu aktuální stav, kategorizovat informace ohledně teorie programovacích jazyků a možností jejich použití při výuce. Práce se zaměřuje na odborné školy, a tak vybraný nástroj má být použitelný nejen ve výuce, ale i v praktickém komerčním světě.

KLÍČOVÁ SLOVA

programování, programovací jazyk, vývoj aplikací, GUI, framework, Qt

ANNOTATION

This thesis is focused on comparing the options of programming languages and techniques for developing applications with graphical user interface. The materials basically sum up current state, organize informations about programming languages theory considering their usage in education. While the thesis is focused on vocational school the chosen toolkit should be useful not only in education, but also in practical commercial life

KEYWORDS

programming, programming language, application development, GUI, Qt

Obsah

1 Úvod

Předmětem této bakalářské práce je problematika výuky programování na středních školách (SŠ) zaměřených na informační technologie a konkrétně se zaměřuje na výuku programování, resp. vývoje aplikací s grafickým uživatelským rozhraním (GUI).

Cílem předkládané práce je vybrat vhodné programovací jazyky, technologie a nástroje pro výuku vývoje aplikací s grafickým uživatelským rozhraním na SŠ a navrhnout modelové lekce pro výuku uvedeného tématu na SŠ.

S ohledem na stanovený cíl práce byl stanoven postup řešení, který spočívá v mapování a analýze řady oblastí souvisejících s problematikou, resp. předmětem práce.

Práce se tak bude věnovat mapování implementace rámcového vzdělávacího programu na středních odborných školách pro obory spadající do oblasti *Informační technologie 18-20-M/01*. Konkrétně pak míře a způsobům, kterým se školy věnují vzdělávání v oblasti vývoje aplikací s GUI.

Dále bude práce analyzovat programovací jazyky, vymezí základní pojmy a stanoví kritéria pro kategorizaci jazyků s ohledem na záměr práce. Na základě těchto kritérií se z širší množiny běžně rozšířených programovacích jazyků vybere menší skupina, u které bude provedena podrobnější analýza a srovnání, které by měly vyústit ve výběr jednoho programovacího jazyka.

Pro vybraný programovací jazyk pak budou podobným způsobem z větší množiny frameworků zvoleny technologie pro tvorbu aplikací s grafickým uživatelským rozhraním. Vybraná technologie, resp. framework pak bude podrobněji popsán s ohledem na možnosti využití ve výuce na SŠ a budou připraveny modelové lekce použitelné ve standardní výuce.

2 Východiska práce

2.1 Rámcový vzdělávací program

Rámcový vzdělávací program (dále jako RVP) oblast programování popisuje následujícím způsobem:

„Cílem obsahového okruhu je naučit žáka vytvářet algoritmy a pomocí programovacího jazyka zapsat zdrojový kód programu. Žák porozumí vlastnostem algoritmů a základním pojmům objektově orientovaného programování, dále se naučí používat zápis algoritmu, datové typy, řídící struktury programu, jednoduché objekty a základní příkazy jazyka SQL. Podstatnou část vzdělávání v programování a vývoji aplikací představuje samostatná tvorba jednoduchých aplikací, statických a dynamických WWW stránek“. [1]

Popis ve srovnání s jinými oblastmi je velmi nespecifický a obsahuje pouze základní informace. Vývoj aplikací se tak v pojetí RVP chápe jako základy programovacích technik (zápis algoritmu, základní řídící struktury, datové typy a úvod do OOP) a pro praktické nasazení je potřeba ho vhodně rozšířit. RVP neuvádí pokročilejší aplikace, jako jsou desktopové programy, složitější konzolové aplikace nebo aplikace se síťovou vrstvou.

2.1.1 Implementace ve školách

První důležitý krok je zmapování situace ve školách. Výběr vychází z pouze z oborů, které jsou odvozeny od oboru *18-20-M/01 - Informační technologie*. Velké množství škol na svém webu nezveřejňuje ŠVP, zveřejňuje pouze obecné informace, ze kterých nelze určit přesný obsah vyučované látky. Záznamy níže jsou tedy ze škol, které ŠVP uvedly, nebo naznačují na svých webech okruhy, kterými se zabývají. Při výběru jsou zohledněny obory, které se týkají programování a vývoje aplikací.

Smíchovská střední průmyslová škola uvádí obecné informace a neuvádí tvorbu aplikací s uživatelským rozhraním (2). Zřizovatelem je hlavní město Praha.

„Cílem je naučit studenty vytvářet algoritmy a pomocí programovacího jazyka napsat zdrojový kód programu.

[1] Rámcový vzdělávací program pro obor vzdělání - 18 - 20 - M/01 Informační technologie. UČEBNÍ DOKUMENTY - Národní ústav odborného vzdělávání [online]. 2018 [cit. 2018-04-15]. Dostupné z: http://zpd.nuov.cz/RVP/ML/RVP%201820M01%20Informacni%20technologie.pdf

Absolvent zná vlastnosti algoritmů, umí analyzovat úlohu a algoritmizovat ji, zapsat algoritmus vhodným způsobem. Je schopen používat základní datové typy, řídící strukturu programu, vytvářet jednoduché strukturované programy.“[2]

SPŠ a VOŠ Písek (3), zřizovatel Jihočeský kraj. Škola dělí studenty podle specializací. Výstupy 3. a 4. ročníku v předmětu “Programování a vývoj aplikací”.

„3. ročník

Zná syntaxi jazyka C a správu paměti aplikace umí zkompilovat aplikaci ze zdrojových kódů.

Umí samostatně navrhnout a zrealizovat aplikaci řešící zadaný problém

4. ročník

Rozumí pojmům třída, objekt a zná jejich základní vlastnosti Použije jednoduché objekty Umí vytvořit formulářovou aplikaci ve Windows Umí vytvořit klient-server aplikaci Rozumí architektuře MVC a dokáže provést refaktoring do této architektury. Umí vytvořit mobilní aplikaci.“[3]

Toto ŠVP oproti předcházejícímu je rozšířeno o programování desktopových aplikací. Zvolená technologie .NET a C# pro tvorbu GUI aplikací nicméně svazuje uživatele na jednu konkrétní rodinu operačních systémů. Podpora platformy .NET pro Linux obsahuje pouze jádro a základní knihovny, nicméně není určena pro tvorbu multiplatformních aplikací s uživatelským rozhraním.

Střední škola informatiky a cestovního ruchu SČMSD Humpolec, s.r.o., soukromá škola zřízená soukromým zřizovatelem, ve svém ŠVP (4) implementuje jazyk Java SE, ve kterém vytváří i grafické rozhraní.

---

[2] Zájemci o studium. Smíchovská střední průmyslová škola [online]. [cit. 2018-04-15]. Dostupné z: http://www.ssps.cz/candidate/read/119

[3] Učební osnovy pro ŠVP: č. j. SPŠ/1380/2014. Smíchovská střední průmyslová škola [online]. [cit. 2018-04-15]. Dostupné z: http://www.sps-pi.cz/wp-content/uploads/2016/12/%C5%A0VP_IT_U%C4%8Debn%C3%ADOsnova_Od_2014-2015_Z02_Web.pdf

Soukromá střední škola výpočetní techniky (5) uvádí ve svých materiálech pro zájemce o studium v rámci předmětu programování navrhování grafického prostředí pomocí technologie .NET. V textu se také mluví o vykreslování pomocí GDI+ knihovny. Uváděný rozsah učiva je největší mezi všemi mapovanými.

Střední průmyslová škola elektrotechniky a informatiky Ostrava má vysoký podíl předmětu Programování (6) (v součtu je to 11 hodin za 4 roky studia). Studenti začínají s používáním jazyka C a přechází ve 3. a 4. ročníku na jazyk C++. Pro tvorbu grafických aplikací používají objektově orientovaný framework VCL, který byl vyvinut společností Borlan původně pro Pascal (Object Pascal) později Delphi. Tento framework (VCL) používá RAD Studio Form Designer[4] a jde o multiplatformní framework. Zajímavá je diskuse na komunitní stránce StackOverflow, kde tazatel žádá o srovnání s frameworkem Qt (viz níže), tato diskuze je z roku 2008.[5]

Závěr z tohoto mapování je, že školy buďto neprogramují vůbec aplikace s uživatelským rozhraním, případně tvoří pouze mobilní aplikace nebo tvoří i desktopové aplikace. V tomto případě tak může stát za úvahu, která použitá technologie je pro výuku a uplatnění absolventů vhodná.

2.1.2 Specifika odborného vzdělávání

Práce se zaměřujeme na odborné vzdělávání, které má podle RNDr. Pavly Zieleniecové CSc. z MFF UK určitá specifika oproti vzdělávání gymnaziálnímu. Zieleniecová ve své prezentaci (6) uvádí mj. následující informace.

„Požadavky na odborné vzdělávání a způsobilosti (kompetence) absolventů vycházejí z požadavků trhu práce popsaných v profesních profilech a kvalifikačních standardech, na jejichž zpracování se podíleli také vybraní představitelé zaměstnavatelů.

…

RVP jsou zpracovány tak, aby zajišťovaly srovnatelnou úroveň odborného vzdělávání a přípravy všech absolventů a aby zároveň umožňovaly škole reagovat na potřeby trhu práce

[4] http://docwiki.embarcadero.com/RADStudio/Tokyo/en/Form_Designer

[5] https://stackoverflow.com/questions/72799/qt-or-delphi-if-you-were-to-choose-one-over-the-other

v regionu, popřípadě prohloubit vzdělávání směrem k určitým skupinám odborných činností.“[6]

Z výše uvedených kritérii je nutné zvážit u vybíraných technologií jejich uplatitelnost do praxe a nezaměřovat se pouze na názornost či didaktickou vhodnost daného jazyka.

2.2 Stav na trhu práce

Ke zmapování situace na trhu práce a využitelnosti jednotlivých programovacích jazyků je obtížné stanovit jednoznačnou metodiku. Trh práce je velmi dynamický a stále se objevují nové technologie a postupy (speciálně v oboru vývoje aplikací).

2.2.1 Rozlišení podle mzdy

Ukazatelem velké poptávky a uplatnitelnosti jsou mzdy, proto je logické zaměřit se na technologie, které jsou žádané. Tato rozvaha je napadnutelná argumentem, že střední škola nebude schopná zajistit vědomosti nebo směřování v dostatečném rozsahu, tedy, že absolvent bude příliš nezkušený pro uplatnění získaných zkušeností v praxi. Zároveň je riziko přílišné zaměřenosti na jeden konkrétní postup a ztráta univerzálnosti jedince. Podle článku na webu mesec.cz (8) z roku 2012 pobírají nejvyšší plat (mzdu) Java vývojáři, kdežto nejmenší platy jsou mezi kodéry HTML nebo vývojáři PHP.

2.2.2 Popularita dané technologie

Lze mapovat množství projektů, které jsou v daném programovacím jazyce napsané, nebo které programovací jazyky jsou nejčastěji na webu vyhledávané. Oba tyto přístupy mají svoje rizika. V případě spočítání množství projektů (řádků) nebude existovat možnost, kde lze zpracovat všechny existující projekty, ovšem data budou zkreslována zaměřením serveru (Například GitHub bude odrážet stav opensource aplikací.). Druhý přístup bude zkreslován tím, že na webu typicky hledají začátečníci, a tak budou data vypovídat, které jazyky jsou populární u začátečníků.

Stack Overflow Trends

[6] Obsah vzdělávání: RVP pro odborné vzdělávání. Výuková média. Pedagogika II [online]. [cit. 2018-04-15]. Dostupné z: https://kdf.mff.cuni.cz/vyuka/pedagogika/materialy/2015%20LS/20150311%20Pedagogika%20II%20-%206%20prednaska%20LS%202014-15.pdf

Stackoverflow, nejznámější fórum pro vývojáře, každý rok zveřejňuje statistiku "Stack Overflow Trends" (7). Statistika ukazuje trendy ve vyhledávání na fóru. Technologie jsou volené podle tzv. TIOBE Index (viz níže). Stack Overflow Trends zkresluje data přesně tak, jak je uvedeno v druhém příkladu. David Robinson to na blogu StackOverflow (8) popisuje takto:

„Measuring developer interest based on Stack Overflow questions isn't perfect: some technologies might inspire more questions among its users than others. But we've found it's a simple measure that gives useful insights into the developer ecosystem. It's especially useful for measuring changes over time: when we see a rapid growth in the number of questions about a technology, it usually reflects a real change in what developers are using and learning."[7]

TIOBE Index

TIOBE je společnost, která jako službu nabízí analýzu softwarových projektů a produktů. Společnost na svém webu uvádí, že za dobu své existence (od roku 2000) zkontrolovala 955 miliónů řádků kódu. Pro srovnání, Linuxové jádro má ve verzi 4. 15 (leden 2018) přibližně 20 miliónů řádků.

U popisu této služby (TIOBE Index) uvádí TIOBE (9), že jde pouze o měřítko popularity na základě vyhledávání v různých vyhledávačích. Metodika, která je při měření používaná, vychází z různých vah při vyhledávání výrazů, seskupování podobných pojmů apod.

2.2.3 Křivka postupu při učení se[8]

Pravděpodobně velmi důležitý je i samotný fakt, jak rychle se dá daným jazykem postupovat. Jazyky, které jsou těžkopádné (ne ve smyslu použitelnosti) a vymykají se běžným mentálním postupům u ostatních programovacích jazyků (například Perl), nejsou do pedagogického

[7] Měření zájmu vývojářů založené na otázkách ze Stack Overflow není ideální: některé technologie mohou více než jiné vyvolávat více otázek mezi uživateli. Zjistili jsme, že jde o jednoduchý způsob, jak nahlédnout do vývojářských systémů. Je to vhodné pro zjišťování změn v čase, pokud vidíme rychlý nárůst otázek o nějaké technologii tak to typicky odráží realitu toho, co se vývojáři učí a co používají. ROBINSON, David. Stack Overflow blog [online]. [cit. 2018-04-15]. Dostupné z: https://stackoverflow.blog/2017/05/09/introducing-stack-overflow-trends/

[8] Z angličtiny "Learning curve"

prostředí vhodné. Zároveň je také vhodné brát ohledy na to, čeho lze s daným jazykem dosáhnout a s jakými požadavky na uživatele.

Bohužel na toto téma neexistuje žádná věrohodná studie, ze které by bylo možné čerpat. Většinou se lze setkat se zdroji které, spíše než uvádět tvrdá data, mají za účel čtenáře pobavit.

Jordan Hudgens (věnuje se školení programátorů) na webu Crondose (10) rozděluje výuku programovacího jazyka na tři teoretické fáze: První fázi (cca 0-300 hodin) nazývá "Learning Liftoff". Uživatel je schopen nakonfigurovat prostředí pro vývoj, naučí se programovací jazyk a základní frameworky a je schopen ho používat.

Z této fáze pak přechází do "Twilight Zone" (cca 300-1000 hodin). V této fázi uvádí, že uživatel řeší často své problémy s dokumentací, protože se s příkazy a konstrukty daného jazyka nesžil natolik, aby přešly do dlouhodobé paměti. V tuto chvíli je uživatel schopen ladit existující aplikace, implementovat části, které nemají tutoriál, který šlo následovat krok za krokem. Velký pokrok v této fázi je i v tom, že se uživatel začíná starat o best-practices[9] v daném programovacím jazyce.

Třetí fáze "The Zone" (1000 a více hodin) je o tom, že student je schopen implementovat vlastní pomocné knihovny, plně kontroluje plynutí kódu v aplikaci a je schopen provádět maximální optimalizaci kódu apod.

Vycházejme z 30 vyučovacích týdnů na škole. Toto číslo se může +- 5 týdnů lišit podle toho, jak v daném školním roce vychází prázdniny, nebo jak má daná škola nastavenou praktickou výuku. Pokud přijmeme následující výpočet, který převedeme na křivku učení tak, jak ji stanovil Jordan Hudgens. 30 týdnů × 2 hodinová dotace × 0, 75 (Hudgens hovoří o celých hodinách) × 2 roky dostáváme 90 hodin zkušeností s programovacím jazykem. Zdvojnásobení hodinové dotace (4 hodiny týdně po dobu dvou let) se dostaneme pouze na 180. Což je stále ve fázi "Learning Liftoff".

[9] z angličtiny: Nejlepší praxe. Jde o popis toho, jak věci nejen udělat, ale jak je udělat pořádně a "hezky"

Proto je vhodné podpořit tradiční výukové metody samostatně vytvářenými domácími projekty, případně zvážit navýšení časové dotace, či protažení výuky na větší množství let během studia.

2.3 Shrnutí východisek

Ve školství

Rámcový vzdělávací program školám poskytuje velkou volnost při výuce programování. Školní vzdělávací programy a obsah výuky se tak velmi liší. Některé školy vyučují pouze základy algoritmizace, plní čistě obsah RVP, jiné školy vytváří komplexní obsah výuky tak, že studenti používají velké množství rozdílných technologií a postupů. Aktuální stav by bylo vhodné zmapovat.

Školy na základě hodinové dotace pouze přímou výukou mohou studenty jen obtíženě přesunout přes hranici „Learning Liftoff".

Mimo školství

Srovnávat popularitu programovacích jazyků však není jednoduché a neexistuje přesná metodika.

3 Programovací jazyky

3.1 Dělení programovacích jazyků

Běžné dělení programovacích jazyků, stejné dělení používá ve svých přednáškách RNDr. Nikola Beneš (11), dělí jazyky podle několika kritérií. Kritéria jsou abstrakce, způsob překladu a spouštění, paradigmata a další kritéria. Běžné programovací jazyky spojují vždy několik kritérií

3.1.1 Dělení podle abstrakce

Jazyky dělené podle míry abstrakce jsou nazývány vyšší a nižší. Toto kritérium popisuje složitost operací ve vztahu k počítači. Nízko úrovňové jazyky (např. Assembler) popisují všechny operace, které jsou v registrech procesoru potřeba. Vyšší jazyky veškeré základní operace vykonávají automaticky a není třeba s nimi explicitně pracovat. RNDr. Nikola Beneš ve své přednášce ze 14. prosince 2016 tvrdí, že vývoj míří spíše k vyšším programovacím jazykům.

V kontextu výuky se tedy zdá (s ohledem na praktické použití ve výuce), že pokud chceme následovat trend, budeme volit vyšší programovací jazyk (C++, Java atd.). Zároveň jsou konstrukty vyšších programovacích jazyků lépe čitelné a pochopitelné pro studenty.

3.1.2 Dělení podle způsobu překladu a spouštění

Při vytváření programu dochází typicky k překladu z jazyka vyššího do jazyka nižšího (případně rovnou do strojového kódu). Rozlišujeme dva typy jazyků: interpretované (Jazyk je postupně interpretován.) a kompilované (Program je jednou přeložen a poté se opakovaně spouští.). Moderní jazyky se snaží oba přístupy spojovat, a tak vznikají různé přístupy ke kompilaci (kompilace do mezikódu, částečná kompilace, kompilace za běhu)

Toto kritérium nehraje při výuce velkou roli. Kompilace, případně interpretace jazyka určuje jeho rychlost při opakovaném používání, nicméně ve výuce nehraje rychlost programů zásadní roli.

3.1.3 Paradigmata programovacích jazyků

Dělení podle stylu programování chápeme v základu na **imperativní** (program chápe jako sadu příkazů) nebo **deklarativní** (program logicky členíme na funkce). Moderní jazyky volí multiparadigmatický přístup, kdy kombinují oba přístupy.

Ve výuce budeme potřebovat oba přístupy pro názornost a simulaci logického uvažování. Zatímco imperativní programování je vhodné z hlediska souslednosti a koherentnosti kódu, při složitějších programech se při složitějších úlohách může stát nepřehledné. Čistě deklarativní programování může vést k zmatení na základě nelineárnosti vytvořeného kódu. Je tedy jasné, že pro výuku bude vhodné použít multiparadigmatický jazyk.

3.1.4 Další vlastnosti programovacích jazyků

RNDr. Nikola Beneš ve své přednášce uvádí ještě 3 druhy programovacích jazyků. Jazyky skriptovací, které jsou vhodné pro psaní skriptů např. pro ovládání počítače, či automatizaci úloh. Ty ve výuce lze využít v mezipředmětové vazbě na další předměty, jako jsou např. operační systémy nebo počítačové sítě. Další druh jsou objektově orientované jazyky, zde autor Nikola Beneš zároveň uvádí i fakt, že moderní jazyky nějakou formu objektového programování obsahují, a v kontextu výuky je tak nutné objektový přístup používat. Poslední je v autorčině přednášce uvedena kategorie jazyků paralelních (Možnost využít

distribuované výpočty v daném programovacím jazyce). Distribuované výpočty nejsou předmětem středoškolské látky a není nutné při výběru jazyka na tuto kapitolu brát ohled.

4 Výběr programovacího jazyka

Pascal

Původně procedurální a případně funkcionální paradigma rozšířilo Delphi i o objektově orientované programování a s prostředím Lazarus[10] lze vytvářet i grafické aplikace. Jako předpoklad pro výběr programovacího jazyka na střední odbornou školu je nejen vhodnost pro výuku, ale i praktická uplatitelnost. Z tohoto důvodu není ve srovnání programovací jazyk Pascal, který byl původně pro výuku vytvořený a dodnes je na některých školách používaný.

Python

Python[11] je multiparadigmatický (procedurální, funkcionální a objektově orientované programování) skriptovací jazyk. V tomto srovnání nebude figurovat také programovací jazyk Python, protože se liší od ostatních jazyků syntaxí a zhoršuje podmínky pro přechod na jiný programovací jazyk. V případě výuky lze nicméně Python velmi dobře zařadit, protože umožňuje vytváření webových aplikací (například pomocí frameworku Django), nebo vytváření grafických aplikací (wxWidgets, PyQt, PySide).

C#

Jazyk C# je vyšší, čistě objektově orientovaný jazyk (12) a velmi často je srovnáván právě s Javou. Jeho původní doménou jsou spíše aplikace psané pro rodinu operačních systémů Windows a je spjatá s aplikační platformou .NET. Podpora běhu na Unixových server/stanicích je přes projekt MONO a v nedávné době i podpora (13) právě .NET frameworku. Správa paměti je automatická a kód je mezi verzemi zpětně kompatibilní.

Velkou výhodou je právě návaznost na produkty firmy Microsoft i implementace ve vývojovém prostředí Visual Studio. Nevýhodou je znovu uvedená návaznost na platformu, nemožnost použití procedurálního nebo funkcionálního programování. Grafické knihovny

[10] https://www.lazarus-ide.org/

[11] https://www.python.org/

nejsou podporovány na jiných platformách. Z propedeutického hlediska nepovažuji automatickou správu paměti za vhodnou, zároveň je Java stále rozšířenější jazyk a nezdá se, že by SE, díky popularitě OS Android, na tom mělo něco měnit.

C

C bylo vytvořeno v roce 1978 a v dalších letech prošlo vývojem. Celý vývoje je dokumentovaný a normovaný, naposledy v roce 2015 normou ISO/IEC TS 18661-2:2015. Jde o vysokoúrovňový programovací jazyk (14), který ale obsahuje prvky i nízko úrovňových operací a je tak velmi často používaný mezi vývojáři mikročipů. V jazyce C můžeme vytvářet i grafické aplikace, například pomocí grafického frameworku GTK. C se rozhodně hodí pro školy, kde se pracuje s elektrotechnikou. C obsahuje pouze datové struktury a nelze tak programovat plně objektově. Z důvodu nepodporování paradigmatu objektově orientovaného programování se nehodí pro výuku obecného programování. Existuje nástavba Objective C, která objektově orientovaný přístup podporuje.

C++

C++ je programovací jazyk, který vysokoúrovňový, multiparadigmatický a typovaný. Podle domovské stránky (15) projektu jde o "lepší C, které podporuje abstrakci dat[12], objektově orientované programování a generické programování[13]". C++ je standardizováno pomocí mezinárodního standardu ISO a aktuální platná norma je ISO/IEC 14882:2017. Jde o jazyk kompilovaný a pro překlad je nutné mít nainstalovaný kompilátor. Zdarma jich je několik a hlavní jsou MSVC (Microsoft Visual C++) od společnosti Microsoft, případně projekt svobodného překladače GCC (Gnu Compiler Collection), který je pro platformu Windows distribuovaný jako projekt MinGW. Většina komponent je licencovaná pod licencí MIT License, nicméně komponenty vycházející z projektu GNU (GNU Make, aj.) jsou licencovány podle původní licence GNU GPL.

[12] V kontextu tříd (class) http://www.stroustrup.com/bs_faq.html#class

[13] Použití šablon (template)http://www.stroustrup.com/bs_faq.html#generic

Java

Java je vysokoúrovňový, objektově orientovaný a typovaný programovací jazyk. Pavel Herout v knize Učebnice jazyka Java uvádí, že Java vznikla v roce 1991 ve společnosti Sun Microsystems na základě jazyka C++, jako jazyk pro vestavěné systémy[14]. Projekt později prorazil na poli webových aplikací (16). K pozdějšímu rozšíření vedlo i rozhodnutí firmy Google použít Javu jako programovací jazyk pro svou nově vznikající platformu Android.

Java se liší od klasických programovacích jazyků tím, jak připomíná docent Pavel Herout ze Západočeské univerzity, že nepřekládá do relativních adres, nýbrž do jazyka "byte-code", který je interpretován na spouštěcím počítači pomocí JVM[15]. Na základě tohoto lze Javu považovat za interpretovaný jazyk.

5 Srovnání Java a C++ s ohledem na výuku

Jak Java, tak C++ jsou moderní programovací jazyky, které jsou velmi rozšířené, aktivně používané a lze je přímo uplatnit na trhu práce. Syntaxi a způsob práce lze přenést na další programovací jazyky na trhu. Následující srovnání porovnává rozdíly práce mezi jazyky samotnými a odchylku od konceptů běžných v programování. Z těchto důvodů se tyto programovací jazyky nejvíce hodí pro výuku programování (a nejen samotné algoritmizace) na střední odborné škole, jejímž cílem je nejen osobní rozvoj, ale i praktická uplatnitelnost po absolvování.

5.1 Základy obou jazyků

Oba jazyky se velmi podobají základní syntaxí. C++ je rozšířením jazyka C (jak je uvedeno na domovské stránce projektu). Docent Pavel Herout v knize "Učebnice jazyka Java" píše, že společnost Sun Microsystems Javu vyvíjela na principech C/C++. Vidět je to i na syntaxi, kde lze nacházet naprosto stejné využití. Cykly, podmínky, deklarace jsou velmi podobné. Lze nalézt drobné rozdíly, například Java neumožňuje v podmínce dosazovat, následující syntaxe může být v C/C++ použita.

```
while ((row = mysql_fetch_row(res)) != NULL)
```

[14] V originálu a obecně IT se používá pojem „Embedded devices"

[15] Java Virtual Machine

5.2 Kompatibilita se systémovými komponentami

Linuxové jádro jako zástupce nejrozšířenějšího operačního systému na světě je z 96 % vytvořeno v C[16]. Proto je pro programovací jazyk důležitý zachovat přístup k systémovým komponentám a knihovnám. C++ je zpětně kompatibilní s C. Java pro nativní volání využívá techniky Java Native Access (JNA)[17] a Java Native Interface (JNI)[18].

5.3 Podporované styly programování

Zatímco C++ podporuje čistě procedurální nebo funkcionální programování, Java je jazyk postavený na principu OOP (Samozřejmě je možné psát procedurální kód, ale nutně uvnitř jedné třídy.). Oba jazyky podporují generické programování. Jak Java, tak i C++ podporují metaprogramování. Na základě tohoto bodu se jeví C++ výhodnější, právě z důvodu přímé podpory procedurálního/funkcionální programování, které může být určité skupině začátečníků přívětivější.

5.4 Překládání a spouštění

C++ a Java se liší ve spouštění, zatímco v obou případech jde o překlad vyššího jazyka na jazyk nižší, C++ se překládá do instrukcí strojového kódu pro danou platformu, Java překládá do "byte-code", který je spouštěn v JVM, což je interpret těchto instrukcí, který je nutné mít na spouštěcí platformě nainstalovaný. Koncept jazyka C++ se označuje jako "Write once compile anywhere" (WOCA), tedy "Napiš jednou, kompiluj kdekoliv". Koncept jazyka Java se označuje jako "Write once run anywere (WORA)", přeloženo jako "Napiš jednou, spusť kdekoliv". Snadné spouštění a přenositelnost funkčních programů je tedy výhodou Javy. Oba jazyky podporují velké množství platforem, které jsou v jednom případě (Java) zajištěné vytvořením JVM pro danou platformu a v druhém případě překladačem pro danou platformu (C++).

---

[16] https://github.com/torvalds/linux, stav 11. března 2018

[17] https://github.com/java-native-access/jna - Veřejný Github repozitář frameworku Java Native Access

[18] https://docs.oracle.com/javase/7/docs/technotes/guides/jni/spec/functions.html - specifikace volání Java Native Interface

Pro Javu existovala podpora v projektu Gnu Compiler Collection[19] do verze GCC 6, kdy bylo možné překládat Javu do nativního strojového kódu. Android používá programovací jazyk Java, nicméně byte code interpretuje pomocí systému Darvik[20] (tudíž nemá klasický JVM) a v novějších verzích přímo do do nativních binárních souborů elf[21].

5.5 Datové typy

Základní datové typy

Oba jazyky podporují běžné základní datové typy (int, double atd.), nicméně Java mezi standardní datové typy počítá i String. Řetězce jsou v C++ reprezentovány třídou string, kterou je nutné vkládat do projektů. Java definuje rozsah datových typů. nicméně v případě jazyka C++ je rozsah závislí na implementaci v překladači.

Typy proměnných

Java rozděluje proměnné na primitivní (int, apod.), referenční (ukazatele na objekt) a tyto jsou předávány do funkcí právě a pouze hodnotou. C++ umožňuje tři způsoby předání (hodnotou, odkazem, nebo ukazatelem pro všechny datové typy). Svoboda může být brána jako výhoda při programování v jazyce C++, ale může vést k plýtvání paměti (zbytečné kopírování objektů do funkcí).

Bezznaménkové datové typy

C++ umožňuje pracovat s nezápornými datovými typy, klíčové slovo je *unsigned*. Java 8 byla první verzí, která práci s bezznaménkovými datovými typy umožňuje[22]. Joe Darcy se na blogu[23] společnosti Oracle zmiňuje a uvádí, že používání bude přes statické funkce primárně v java.lang.Integer a java.lang.Long[24]

---

[19] https://gcc.gnu.org/wiki/GCJ - Domovská wiki stránka projektu

[20] https://source.android.com/devices/tech/dalvik/ - popis projektu Darvik

[21] https://github.com/dogriffiths/HeadFirstAndroid/wiki/How-Android-Apps-are-Built-and-Run Schéma sestavování aplikací v Androidu

[22] http://www.oracle.com/technetwork/java/javase/8-whats-new-2157071.html

[23] https://blogs.oracle.com/darcy/unsigned-integer-arithmetic-api-now-in-jdk-8

[24] https://docs.oracle.com/javase/8/docs/api/java/lang/Integer.html#divideUnsigned-int-int- - dokumentace dělení celých čísel bez znaménka.

5.6 Funkce

Globální funkce

Java je objektově orientovaný jazyk a neumožňuje funkcionální programování (všechny funkce musí být metody nějaké třídy). C++ umožňuje vytvářet funkce v globálním jmenném prostoru. Stejné pravidlo platí i pro globální proměnné.

Výchozí hodnoty

V C++ můžeme definovat výchozí hodnoty parametrů ve funkcích pomocí operátoru přiřazení. Takto definované parametry nazýváme jako nepovinné. Java nutí uživatele do opakované deklarace funkcí.

Přístup definování výchozích hodnot je spíše bližší dalším programovacím jazykům, jako například Python, PHP, aj.

Anonymní funkce a typy parametrů

Java i C++ umožňují předávat (oba jazyky používají rozdílnou terminologii i syntaxi) funkce, třídy. V obou jazycích lze definovat i anonymní funkce (C++ 11 a Java 8).

5.7 Objektově orientované programování

V případě obou jazyků lze definovat programátorské třídy.

Abstraktní třídy

Pokud je nutné vytvářet vzory pro dané třídy, používá Java termín interface a pomocí implements uvádíme, které rozhraní chceme využít. Překladač bude vyžadovat definování všech metod ve třídě, která je definovaná jako Interface. C++ pro dosažení stejného efektu používá abstraktní třídy.

Dědění

Součástí konceptu OOP je i dědění, Java i C++ dědění umožňují. C++ na rozdíl od Javy umožňuje vícenásobné dědění (více předků, než pouze jeden). Tento koncept se často nazývá *mixins.* Java podporu přidává přes použití více rozhraní[25]

Přístup k objektům

Zde narážíme na další rozdíl v jazycích Java a C++. Java přistupuje k objektům pouze na úrovni reference, C++ umožňuje jak přístup přes ukazatel (chápejme jako referenci), tak přístup přímý.

Oprávnění

Oba jazyky implementují ve třídách opravnění private, public a protected. Java označuje metody ve výchozím stavu jako public, kdežto C++ je považuje za private.

Přetěžování operátorů

C++ na rozdíl od Javy umožňuje přetížit operátory, tedy dát význam výrazům jako sčítání dvou objektů dané třídy. Sčítání objektu a proměnné typu int, atd. Java přetěžování objektů neumožňuje.

C++ umožňuje definici operátorů i na úrovni globálních funkcí. Což je logické vzhledem k tomu, že Java nedefinuje nic jako funkce.

Konstruktor a desktruktor

Oba jazyky obsahují konstruktor (metoda třídy volaná při vzniku objektů). C++ navíc obsahuje i desktruktor, který je volán při zaniknutí objektu a slouží například k úklidu paměti, nebo uzavírání souborů, apod. V Javě destruktor neexistuje, protože uvolňování paměti řídí Garbage Collector[26]

---

[25] http://hannesdorfmann.com/android/java-mixins

[26] Existují techniky, které se toto snaží řešit pomocí přetěžování metody finalize, obecně jde o nedoporučovaný postup. Viz odkaz.

https://stackoverflow.com/questions/36319571/how-to-ensure-method-is-called-before-object-is-destroyed

Proč vůbec teoreticky finalize používat proběhla diskuze na webu stack overflow.

https://stackoverflow.com/questions/158174/why-would-you-ever-implement-finalize

5.8 Správa paměti

Správa paměti je jeden z největších rozdílů mezi C++ a Javou. Java používá "Garbage Collection", což je malý program hlídající aktuálnost alokované paměti a v případě, že alokovaná paměť není používaná, paměť uvolní zpět operačnímu systému. V tomto případě je tak nutné, aby Garbage Collection běžel na pozadí a uklízel paměť.

C++ používá explicitní volání funkcí new a delete, kterými lze alokovat nebo uvolňovat paměť dynamicky. Výhoda takového fungování je v nenáročnosti a rychlosti. Špatnou prací s pamětí je možné vytvářet tzv. memory leaky (úniky paměti). Standard C++ 11 implementuje[27] ABI pro Garbage Collection.

5.9 Standardní knihovna

C++ standardní knihovna[28] je zaměřená na základní funkce a navíc přidává pouze podporu překladů, diagnostiky, řetězců, mj. i standardní C knihovny a některé další. Standardní knihovna jazyka Java[29] je daleko rozsáhlejší a její obsah se zvětšuje s každým vydáním. Kromě běžných funkcí obsahuje i tvorbu uživatelských rozhraní, podporu pro různé datové formáty (XLS, XML, aj.), webové služby, tiskové služby a jiné.

5.10 Ochranné známky

Java je na rozdíl od označení C++ vedena jako ochranná známka. Její registrátor je společnost Oracle, která původního vývojáře (Sun Microsystems) odkoupila. U registrace[30] ochranné známky jsou uvedené všechny "Goods" (zboží), které s ochrannou známkou souvisí.

Při výpisu[31] z databáze Úřadu průmyslového vlastnictví (ČR) je vidět, jak se registrace vyvíjela, původní vlastník byla výše zmíněná společnost Sun Microsystems.

---

[27] http://www.stroustrup.com/C++11FAQ.html#gc-abi

[28] http://en.cppreference.com/w/cpp/header

[29] https://introcs.cs.princeton.edu/java/stdlib/

[30] Vyhledávací systém úřadu pro patenty a obchodní značky USA - http://tmsearch.uspto.gov/bin/gate.exe?f=doc&state=4805:en6kdg.31.235

[31] Výstup z databáze úřadu průmyslového vlastnictví

5.11 Závěr

Výběr programovacího jazyka z dvojice Java a C++ je poměrně složitý. Vždy je nutné zamyslet se nad kontextem, ve kterém se chceme pohybovat. Java umožňuje jednodušší psaní (absence práce s pamětí), její silnou stránkou je nativní podpora v Androidu, což může hrát významnou motivační roli a faktickou uplatnitelnost na trhu práce. Java ulehčuje práci s pamětí, ale správně pochopená práce s pamětí v C++ je uplatnitelná i v dalších jazycích.

Výhoda C++ je v tom, že velké množství programovacích jazyků je jeho syntaxí ovlivněna a pochopení koncepce tak vede k lepšímu porozumění při práci s dalšími programovacími jazyky. C++ v kombinaci s vhodně zvolenými frameworky (gtkmm, Qt) a pestrou sadou knihoven může být použit pro tvorbu libovolných aplikací, včetně aplikací grafických, které (jak je zmíněno u Javy) hrají velikou roli při motivaci při výuce programování.

V této práci se budeme zabývat programovacím jazykem C++ a jeho aplikací při praktické výuce programování na střední odborné škole, typicky vhodné pro obor Informační technologie (18-20-M/01). Při výuce je vhodné zvážit, jestli nezařadit předmět "Vývoj mobilních aplikací", nebo alespoň předmět "Programování v Javě", protože Java je stále velmi rozšířený jazyk. Podle TIOBE index z března 2018 je dokonce 1. v indexu popularity, ačkoliv již dlouhodobě stagnuje. C++ se pro srovnání v uplynulých letech drží s drobnou stagnací na 3. místě tohoto měření. Pro zajímavost je dobré uvést, že jazyk C v posledních letech výrazně vyrostl (pravděpodobně z důvodu popularity IoT[32])

6 Tvorba aplikací s grafickým uživatelským rozhraním v jazyce C++

6.1 Předpoklady pro práci s GUI frameworky

Tvorba grafických aplikací v C++ vyžaduje základní znalost algoritmizace a základní pochopení objektově orientovaného přístupu při tvorbě aplikací. Student musí znát základní konstrukty jazyka.

https://isdv.upv.cz/obr/ozvyprej/249/O-103249.pdf

[32] Internet of Things - Internet věcí

V jazyce C++ je potřeba znát a dobře pochopit správu paměti a předávání do funkcí (předávání hodnotou, odkazem, referencí). Techniky je dobré opakovat při výkladu teorie tvorby aplikací grafických aplikací a upevňovat opakováním.

Z objektově orientovaného přístupu je nutné znát dědění, chápání koncepce tvorby tříd a objektů. Výhodné je také rozumět pojmům konstruktor, destruktor.

6.2 Výběr grafického frameworku

Pro tvorbu grafických aplikací v jazyce C++ je na výběr z množství knihoven a frameworků. Knihovny se liší podporovanými platformami, použitou licencí, vhodnými vývojovými nástroji a v neposlední řadě také existující dokumentací a knihovnou. Některé grafické frameworky mají vytvořené i návrháře, což jsou aplikace použitelné k vizuálnímu návrhu aplikace a umožňují tak velmi jednoduchou tvorbu uživatelského rozhraní.

Grafické aplikace sdílí obecné principy (například smyčka událostí apod.), nicméně v aplikaci se liší. Na následujících řádcích představím grafické knihovny, které jsou více rozšířené a je možné je využít při tvorbě aplikací s grafickým uživatelským rozhraním. Základní předpoklad pro výběr knihovny je běh na různých platformách.

6.2.1 WxWidgets

WxWidgets projekt vznikl (17) v roce 1992 jako univerzitní projekt multiplatformní knihovny, která bude podporovat jak Windows, tak i Unixová zařízení. Původní autor se jmenuje Julian Smart a framework vyvíjel na univerzitě v Edinburgu.

Podporované jazyky a platformy

Knihovna je původně psaná v C++, ale podporuje i další programovací jazyky[33]. Nativní podpora je jazyka C++ a volání v ostatních knihovnách jsou na C++ překládané. Existuje podpora pro například: Javu, Perl, Rust, PHP, Python, aj. Knihovna nabízí několik různých vydání a linkováním odpovídajícího vydání lze emulovat vzhled dané platformy (Windows, MacOS, Linuxové platformy přes knihovnu GTK, X11 aj.).

[33] https://wiki.wxwidgets.org/Bindings

Implementované funkce

WxWidgets mají v popisu uvedeno, že poskytují API ke snadnému použití při tvorbě aplikací s grafickým uživatelským rozhraním a navíc poskytují podporu pro tvorbu síťových aplikací, streamů, podporu pro schránku[34], drag and drop[35]. Pomocí WxWidgets lze vytvářet vícevlákenné aplikace, zpracovávat obrázky, pracovat s HTML. Implementována je podpora multimediálních prvků, jako jsou video/audio přehrávače.

Licencování

WxWidgets jsou licencované pod vlastní licencí wxWindows License, která je založená na L-GPL, ale není natolik přísná. Obsahuje výjimku, která nevyžaduje uvolňování zdrojových kódů, ale vybízí k uvolňování kódů úprav v samotné knihovně.

Rozšíření

WxWidgets uvádí, že má 1800 aktivních odběratelů novinek. Nejznámější projekty vytvořené pomocí wxWidgets jsou uvedené AudaCity, AVG AntiVirus, Forte Agent, Filezilla, iPodder a Tortoise CVS. Organizace používající tento framework jsou uvedeny: AOL, AMD, Lockheed Martin, Xerox, NASA, OSAF (Open source applications foundation).

Podpora ve vývojových prostředích

Podle oficiální wiki stránky projektu wxWidgets jsou uvedená vývojová prostředí pro vývoj aplikací s wxWidgets, nejvýše je uvedené multiplatformní IDE Code::blocks, které je pomocí wxWidgets také vytvořené. Jako další možnosti jsou uvedené editory jako NetBeans, Eclipse, Microsoft Visual C++ (pouze platforma Windows), a další.

6.2.2 Qt

Historie Qt podle oficiálních materiálů (18) začíná v roce 1990 na lavičce v Norsku, kde Haarvard Nord a Eirik Chambe-Eng vytvořili první koncept Qt[36]. První vydání přinesla společnost TrollTech 20. května 1995 jako verzi 0.90 pro Linux pod licencí “Commercial &

[34] ang. clipboard, paměť používaná pro kopírování a vkládání

[35] Táhni a pusť. Přenášení dat pomocí přetahování myší

[36] Qt se v angličtině vyslovuje jako “CUTE”, což v překladu znamená roztomilý

open source (FreeQt license)". První zákazníkem se stala Evropská kosmická agentura a ve stejném roce (1996) se stala knihovna Qt podkladem pro vývoj linuxového desktopového prostředí KDE, Qt je garantováno jako svobodná knihovna (s určitými omezeními, která z použité licence plynou). Tento rok také je také vydána verze 1. 0.

Významný rok byl 2008, kdy finská společnost Nokia provedla akvizici společnosti TrollTech (původní vlastník a vývojář Qt) a začíná pro Qt používat značku Qt Software at Nokia. O rok později, v roce 2009, je vydáno IDE Qt Creator, které je zaměřené čistě na tvorbu Qt aplikací. Další roky byly doplňovány nové technologie (podpora WebKit, sada nástrojů QtQuick). Qt se stalo vývojářským nástrojem pro Symbian.

Od roku 2011 předává společnost Nokia (během svého kritického období, které později vedlo k faktickému krachu společnosti) nejprve komerční licence a později všechna práva do společnosti Digia. Následuje přepis celé programové základny, vychází verze Qt 5.0 a Digia se přeměňuje na "The Qt Company". Během roku 2015, kdy se oslavovalo 20. výročí prvního veřejného vydání Qt, uvádí projekt více než 800 000 aktivní uživatelů na celém světě.

Podporované jazyky a platformy

Qt je multiplatformní projekt, který jako podporované platformy uvádí desktopové Windows, Linux, Mac OS, mobilní, ať už známější (Android, iOS, Windows Phone) nebo méně rozšířené (BlackBerry, Ubuntu Touch, Sailfish OS). V poslední době se Qt snaží zapojit do vývoje pro vestavěné systémy[37].

Hlavní podporované jazyky jsou C++ a QML, což je specifický jazyk pro návrh a tvorbu uživatelských rozhraní. Další podporované jazyky (třetími stranami) jsou Python, Rust, Go, Node.js, Java, C# a další.

Implementované funkce

Qt podporuje v základu tvorbu aplikací z uživatelských aplikací a zakládá si na nativním[38] vzhledu v každém prostředí. K vykreslování nicméně nepoužívá knihovny dané platformy,

[37] https://www.qt.io/qt-for-device-creation/

[38] Přirozeném

ale vše implementuje sama o sobě. Projekt Qt postihuje velkou část tvorbu aplikací. Lze vytvářet nejen uživatelské rozhraní, ale přes různá rozhraní lze používat 2 D/3 D grafiku, pracovat s databázemi (různé typy), převádět data ze sériového portu, zobrazovat webové stránky, používat Bluetooth, zpracovávat multimédia, vytvářet komplexní síťové aplikace, apod.

Qt používá při sestavování systém qmake, který překládá Qt projekty na standardní C++ make projekty a ty pak sestavuje pomocí běžně dostupných nástrojů.

Licencování

Licencování Qt je tzv. dvojí. Společnost "The Qt Company", která je zároveň vlastníkem ochranných značek Qt, uvádí dva typy licencí: Opensource a komerční. Opensource licence je GPL, která zavazuje program vytvořený pomocí Qt distribuovat včetně zdrojového kódu, tato licence je garantovaná smlouvou mezi Qt Company a KDE Free Qt Foundation. Komerční licence takové omezení nemá a zároveň nabízí některé další produkty (speciálně podpora vývoje embedded zařízení).

Rozšíření

V roce 2015 se Qt chlubilo zájmem 800 tisíc vývojářů. KDE (desktopové prostředí) je postavené na Qt a uvádí aktivních 1800 vývojářů. Webové stránky (qt.io) a všechna fóra mají vlastní CSS styl a jsou profesionálně vedená. Qt bylo společností Canonical použito jako hlavní framework pro jejich platformu Ubuntu Touch. Další známý projekt je VLC Media Player, nebo Oracle Virtual Box. Ke Qt sáhla i společnost Adobe v produktech Adobe Photoshop Album, nebo Adobe Photoshop Elements (20)

Podpora ve vývojových prostředích

Qt používá při sestavování nestandardní postup pomocí qmake a zavádí do standardu jazyka C++ nová klíčová slova. Tato klíčová slova by běžný editor označoval za chyby, a tak je velmi často pro plnou a bezchybnou podporu Qt potřeba doinstalovat nějaký plugin. Například pro Visual Studio existuje rozšíření Qt Visual Studio Tools[39].

[39] https://marketplace.visualstudio.com/items?itemName=TheQtCompany.QtVisualStudioTools-19123

Pravděpodobně nejlepší možností je používat Qt Creator, což je vývojové prostředí, které je vytvořeno právě pomocí knihovny Qt a jeho kód je otevřený[40]. Qt Creator v komunitní verzi je možné používat zcela zdarma, právě pod výše uvedeným licenčním ujednáním (resp. omezením). Qt Creator je rozsáhlé IDE[41], které mimo jiné obsahuje i program Qt Designer, jenž umožňuje návrh aplikací pomocí jednoduchého grafického editoru, integrované podpory systému verzování Git, debugování, apod.

6.2.3 Další grafické frameworky

GTK+

GTK+ je další grafický framework, který podporu pro C++ integruje v projektu gtkmm, nicméně podpora vývoje v prostředí Windows je minimální. Framework sice funguje multiplatformě, nicméně vývoj se zaměřuje hlavně na Linux[42]. Výhodou tohoto frameworku je psaní nativního ködu C++ a překlad pomocí standardních nástrojů překladu (jako je například make).

Windows Forms API

Windows Forms API, jak již název napovídá, je grafický framework vydávaný společností Microsoft s integrovanou podporou ve Microsoft Visual Studio. Microsoft nicméně pro vytváření nativních aplikací podporuje spíše C# a podporu C++ je potřeba povolit[43]. Výhodou oproti všem ostatním frameworkům je nativní vzhled aplikací, nevýhodou je ne zcela intuitivní API a absence multiplatformnosti.

Zvolený grafický framework

Pro vývoj je potřeba vždy zvážit konkrétní potřeby dané aplikace a vybrat tak grafický framework. Pro potřeby výuky zvolíme Qt. Hlavní výhody jsou podpora více platforem, včetně vývoje pro Android, integrovaný návrhář grafických rozhraní a plnohodnotné IDE,

---

[40] https://github.com/qt-creator/qt-creator

[41] IDE - Integrated Development Environment, Integrované vývojové prostředí, sada nástrojů nebo programů, které vývojář používá při vývoji programu

[42] https://www.gtkmm.org/en/download.html

[43] https://social.msdn.microsoft.com/Forums/vstudio/en-US/e6fbde42-d872-4ab3-8000-41ab22a4a584/visual-studio-2017-windows-forms?forum=winformsdesigner

které po instalaci umožňuje plnohodnotné vytváření C++/Qt aplikací. V tomto IDE lze vytvářet i čistě C nebo C++ aplikace pro příkazovou řádku. Většina škol bude samozřejmě používat ten grafický framework, k jehož výuce má dostupné odborníky, nebo dostatečně proškolené učitele.

7 Výuka tvorby grafických aplikací v Qt, C++

7.1 Předpoklady

Nároky na pomůcky

Učebna s odpovídajícím množstvím počítačů s nainstalovaným programem Qt Creator. Pro názornost výuky je vhodné použít dataprojektor (pro ukázky programů) a tabuli, kde lze ilustrovat programy, nebo znázorňovat vývojové diagramy. Tabule by neměla být sdílená s dataprojektorem jako plátno, znesnadňuje to výklad. Při vývoji databázových aplikací je nutné mít nainstalovaný nějaký program pro práci s databází (pro výukové účely a jednoduchost není vhodné čistě SQL konzole). K práci s SQLITE lze použít například Sqlite Browser.

Programátorské dovednosti

Předpokládejme, že studenti ovládli objektově orientované programování na alespoň základní úrovni, funkcionální programování jim nedělá syntaktické problémy a s jistotou píší řídící konstrukty, jako jsou podmínky (včetně switch podmínek), řídící cykly apod.

Jazyková výbava

Je velmi vhodné v mezipředmětové spolupráci s AJ pracovat na systematické tvorbě slovní zásoby a gramatické výbavy pro schopnost číst a porozumět anglickému zdroji na zadané téma. Tímto způsobem v kombinaci s dokumentací lze vhodným formováním dosáhnout vysoké míry samostatnosti při tvorbě vlastních programů.

7.2 Konfigurace vývojového prostředí

7.2.1 Instalace Qt Creator

Pro výuku je potřeba nainstalovat a zprovoznit Qt Creator. Je maximálně vhodné proces instalace nějakým způsobem zaznamenat (video, prezentace) tak, aby studenti mohli docílit v ideálním případě stejné konfigurace jako ve škole.

Instalační soubor lze stáhnout na oficiálních stránkách projektu (https://www.qt.io/) v sekci "Download. Try. Buy.". Zde je na výběr mezi komerční a komunitní verzí. Komunitní verze je svázána (viz výše) GNU GPL licencí, nicméně je zcela zdarma a funkčnost je zcela identická až na absenci podpory embedded zařízení.

Instalace QT Creatoru probíhá typicky prostřednictvím programu, kde uživatel volí, které komponenty potřebuje. Pokud navolí všechny, může se dostat až na 200 GB. Komponenty jsou různého typu, ať už jde o překladače pro různé platformy (Windows, Windows Phone, Android, aj.), tak různé rozšíření Qt (jádro prohlížeče Chromium). Vybráním verze Qt se nainstaluje komponenta QMake, která slouží k převádění Qt projektů do Makefile souborů.

Výběr překladače pro Qt

Ve Windows lze zvolit dva základní překladače – MinGW a MSVC[44]. MinGW je převedení projektu Gnu Compiler Collection a umožňuje používání make i v prostředí Windows. Dosažení tohoto efektu je převedením běžně používaných Unixových utilit do Windows kompatibilní podoby s použití standardních knihoven libc++ a dalších. MinGW je populární hlavně u projektů, které programují Linuxový vývojáři a potřebují použít křížovou kompilaci[45]. MSVC v posledních verzí začalo zobrazovat chybová hlášení v jazyce systémů, což s nekompatibilitou znakové sady Qt ztěžuje čtení chybových hlášení překladače.

Desktopové aplikace pro Windows typicky používají spíše MSVC, příkladem je třeba webový prohlížeč Firefox (19), Google Chrome MSVC teprve nedávno opustil (20).

[44] Microsoft Visual C++

[45] https://mingw-w64.org/doku.php

Časté problémy

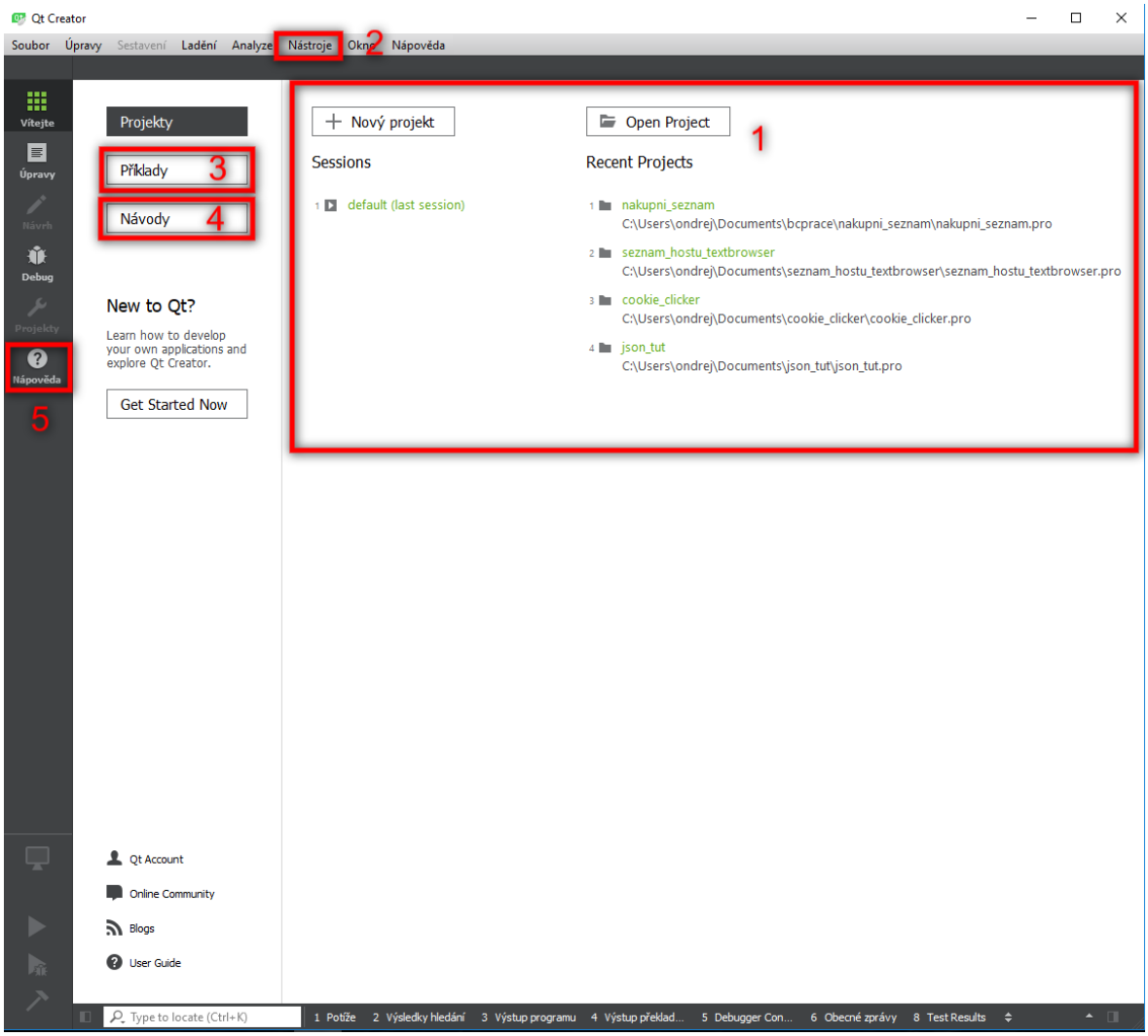
qmake (příkaz pro sestavování projektů) má problém s diakritikou a mezerami v cestě k projektům a je nezbytně nutné projekty sestavovat v adresářích, které nemají diakritiku a mezery v názvu. Ačkoliv bug report pro tento problém je uzavřený, problém stále (21) přetrvává.

Pokud se vyskytne problém s našeptáváním prvků uživatelského rozhraní (ui-> …), je potřeba provést změnu v souboru s uživatelským rozhraním (.ui) a uložit ho. Tak dojde k překompilaci a vytvoření hlavičkového souboru, který Qt používá pro našeptávání.

7.3 Popis IDE Qt Creator

Qt Creator je vývojové prostředí pro C nebo C++ s doplněním o Qt knihovny a technologie. Vývojové prostředí pracuje s projekty, konfiguraci ukládá do souborů .pro, které jsou vlastně direktivy pro qmake.

Vývojové prostředí integruje mimo jiné nápovědu (Obrázek 1.5), správu verzí GIT (vhodné použít při výuce skupinové práce), grafické návrhové prostředí (velmi jednoduchá tvorba uživatelského rozhraní intuitivní formou). Přímo uvnitř jsou integrované návody (Obrázek 1.4), jak například rozhýbat projekt aj. Velká výhoda jsou přímo integrované příklady, které lze otevřít a sestavit přímo uvnitř samotného IDE (Obrázek 1.3).



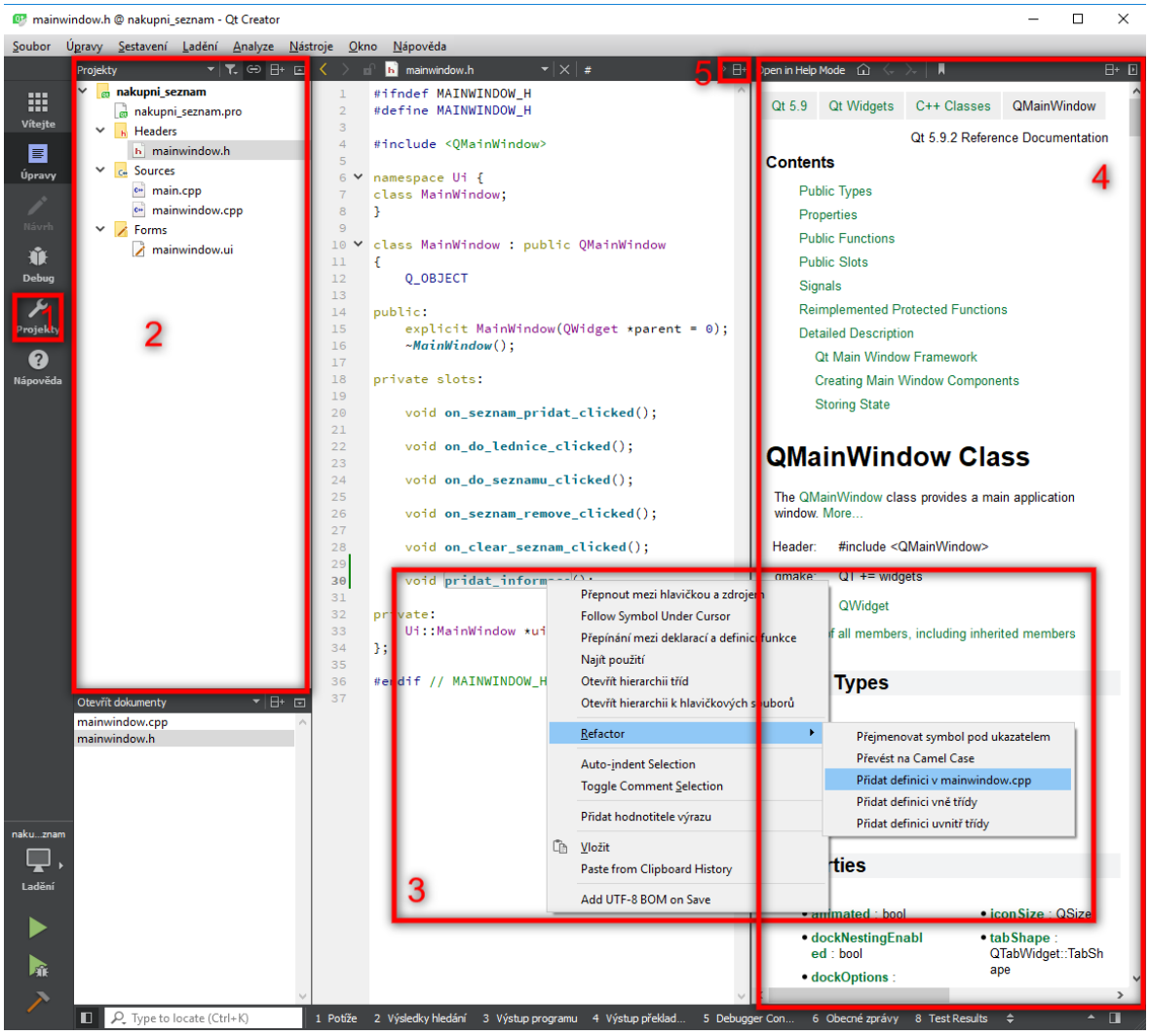
Obrázek 1 Náhled na hlavní okno programu

Rozhraní při práci s projekty

Pokud založíme nový projekt nebo otevřeme starý (projekt typu "Program využívající Qt Widgety), objeví se v levém bočním panelu možnost přepínat mezi režimy (Obrázek 1.1). IDE vytvoří již předem některé třídy a další potřebné soubory včetně .ui souboru. Soubor .ui je soubor editovatelný v režimu "návrh", což je grafický návrhář uživatelského rozhraní.

Nastavení sestavovacích nástrojů, adresářů pro sestavení se aktivuje v kartě pohledy (Obrázek 2.1). Projektové okno (lze přepnout na některý jiný druh pohledu) umožňuje přepínat mezi projekty a soubory aktivního projektu. Soubory řídí podle typu souboru (Obrázek 2.2). IDE integruje pokročile nástroje, jako je například refactorging (na obrázku 2.3), doplňování kódu atd. Kontextovou nápovědu (Obrázek 2.4) lze zobrazit nastavením

kurzoru na nějakou třídu (nebo objekt) a zmáčknutím klávesy F1. Rozhraní celého vývojového prostředí lze dělit na menší části. (Obrázek 2.5). Spouštění aktivního projektu se ovládá vlevo dole na hlavní obrazovce, nebo klávesovou zkratkou (Ctrl + R).



Obrázek 2 Projektové rozhraní

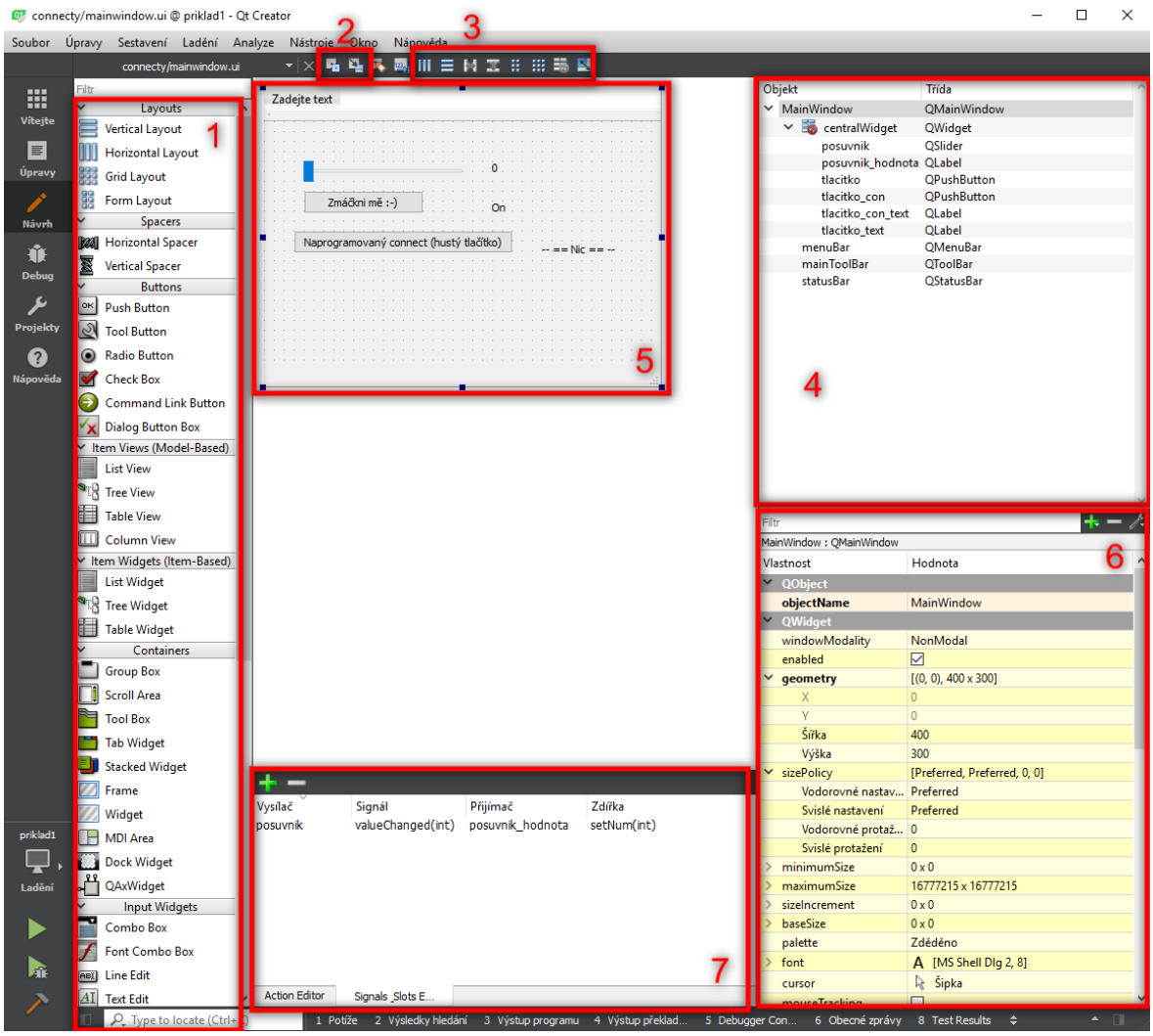
Návrhář

Soubory uživatelského rozhraní se definují v .ui souboru. Jde o XML soubor, který je pouze editovatelný přes interaktivní část aplikace (přímou editace IDE neumožňuje, ruční zásah by mohl ohrozit koherentnost a validitu XML kódu). Z praktických zkušenost lze tohoto návrháře označit za jeden z nejlepších návrhářů integrovaný do IDE. Princip je celkem jednoduchý, hlavní okno aplikace je uprostřed (Obrázek 3.5) a do něj se vkládají prvky uživatelského rozhraní (Obrázek 3.1). Hlavnímu oknu lze nastavit nějaký layout, tzn. jestli

jsou prvky volně ukládané, nebo jsou-li ukládány do nějaké fixní mřížky (Obrázek 3.3). Widgety tak mají strukturu rodič, potomek a takto se to vykresluje do stromu (Obrázek 3.4). Každý widget lze nějakým způsobem přizpůsobovat (způsob zobrazení, velikost, roztahování se aj.). Tyto vlastnosti jsou rozdělené barvou podle třídy, ze které pochází (Obrázek 3.6).

Přímo v návrháři lze propojit již některé signály a sloty. Režim práce v UI lze přepnout (Obrázek 3.2) a propojovat prvky napřímo (Lekce 1. propojení posuvníku a QLabelu). Propojení prvků lze přidávat, upravovat a odebírat v editoru signálů a zdířek (Obrázek 3.7) Pravým na grafický prvek lze aktivně vytvořit zdířku pro implementování vlastní funkce

(Pravý na widget → Jít na zdířku), což vytvoří záznam v hlavičkovém a zdrojovém souboru třídy.



Obrázek 3 Návrhář UI v Qt Creatoru

Hlavní okno aplikace

Každý projekt má hlavní okno. Toto hlavní okno je třída, která dědí od třídy QMainWindow, a do ní se dosazuje jeden objekt třídy QWidget (do kterého lze vložit některé rozvržení a přes něj více dalších prvků). Třídám, které reprezentují okna, lze vygenerovat uživatelské

rozhraní. Ve výchozím stavu se deklaruje třída MainWindow uvnitř jmenného prostoru Ui a tam se v konstruktoru vloží[46] vygenerovaný hlavičkový soubor vzniklý z .ui souboru[47].

Hlavní okno aplikace vytvoří při běžném nastavení i lištu tlačítek, stavovou lištu a lištu s menu. Tyto prvky nejsou nutné a lze je smazat.

7.4 Práce a příklady Qt ve vývojovém prostředí Qt Creator

Knihovna Qt má dobře zpracovanou dokumentaci (22), ve které se lze snadno orientovat, při výuce je vhodné ji se studenty procházet a pomáhat jim orientovat se.

Obecné předpoklady pro všechny lekce

Lekce jsou tematicky zaměřené na jeden typ grafického prvku nebo postupu. Prvotní lekce jsou nezbytné pro pochopení teorie knihovny, další je pak možné různě prolínat, i když řazení odpovídá obtížnosti. Všechny lekce vyžadují alespoň teoretické znalosti z programování, včetně objektového paradigmatu.

Některé lekce navazují na teorie programování, které bude nutné připomenout. Lektor musí zvážit, jde-li o pokročilejší programátory, kteří ovládají C++ s přehledem, nebo o začátečníky. Tomu je nutné přizpůsobit teoretický základ lekce.

7.4.1 Signály a sloty (zdířky)

Základním konceptem Qt jsou signály a zdířky, signály reprezentují události v uživatelském rozhraní (klávesa zmáčknuta, tlačítko stisknuto, myší přetaženo) a sloty[48], což jsou funkce reagující na zadané signály. Spojení signálů a slotů se dělá pomocí funkce `connect`, která je definovaná ve třídě `QObject` a ke své správné funkčnosti vyžaduje v hlavičkovém souboru třídy makro `Q_OBJECT` (viz níže). Pokud chceme některou funkci označit jako slot, je nutné v hlavičkovém souboru třídy použít klíčové slovo (stejně jako se používají režimy oprávnění) `slots`. V následujícím zdrojovém kódu je tento příklad vidět.

---

[46] Toto zajišťuje volání funkce `ui->setupUi(this);` která je definovaná v hlavičkovém souboru uživatelského rozhraní

[47] Pokud nezměníme pojmenování souborů, tak ze souboru mainwindow.ui se vytvoří hlavičkový soubor ui_mainwindow.h

[48] Český překlad uvedený jako "zdířky" se mi nezdá jako příliš štastný, proto budu používat termín "slot" a v plurálu jako "sloty".

```
class MainWindow : public QMainWindow
{
    Q_OBJECT
public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

public slots:
    void onButtonClicked();

...
```

Propojování signálů a slotů

Propojení signálů a slotů lze udělat třemi způsoby.

Graficky v návrháři

V návrháři lze toto propojení implementovat pouze tak, že přenáší stav (stav, který se předá při vyvolání signálu, například změna hodnoty) a neumožňuje žádné výpočty. Propojování lze definovat stiskem klávesy F4, na horní liště návrháře tlačítkem “Edit Signals/Slots”. Pak pomocí techniky táhni a pusť přetáhnout zdroj signálu na cíl. Dialogové okno umožní vybrat na levé straně signál a na pravé zdířku

Pro správu signálů a zdířek lze přepnout na kartu “Signals_Slots Editor” a tam lze existující spojení odebrat a nebo, velmi podobným způsobem jako předešlý, přidat nový signál

Vybrání signálů v návrháři a implementace ve zdroji

Při návrhu uživatelského rozhraní lze jednoduše klepnout na vybraný widget pravým tlačítkem myši a vybrat “Jít na zdířku / Go to slot”. Tento způsob vytvoří slot uvnitř třídy, ke které je .ui soubor vytvořený a přepne tam. Na tomto místě je možné okamžitě začít psát kód.

Pokud bude smazán grafický prvek, je nutné smazat i kód uvnitř třídy. Pokud je prvek přejmenován, je nutné otestovat, jestli spojení je i nadále funkční a případně ho odpovídajícím způsobem upravit.

Manuální propojení pomocí funkce connect

Princip propojování je uvnitř Qt na principu funkce connect. Tento způsob je skrytě zapracovaný i v předcházejících příkladech, nicméně při práci s frameworkem se objeví

situace, kdy nelze ani částečně grafickou cestu použít (například práce se síťovými sockety). Příkaz connect má 4 parametry a v principu jde o zdroj signálu (zdroj), funkce reprezentující signál (signál), kontext volání slotu (cíl) a funkce slotu (slot). Connect je umístěn typicky v konstruktoru, aby k propojení došlo při zapnutí programu.

Níže vidíte příklad volání funkce connect tak, aby došlo k zavření celého okna. Zároveň jsou uvedeny oba způsoby. První řádek zobrazuje modernější syntaxi a druhý původní syntaxi s využitím maker.

```
connect(ui->pushButton, &QPushButton::clicked, this, &MainWindow::close);
connect(ui->pushButton, SIGNAL(clicked(bool)), this, SLOT(close()));
```

Signály a sloty, příklady

Lekce 1

Časová dotace, příprava:

10 minut, jen si vše vyzkoušet.

Trvání:

1 vyučovací hodina

Vstupní požadavky na žáka:

Žáci znají a ovládají objektové paradigma.

Způsob výuky:

Frontální výuka podpořená samostatnou prací s dopomocí lektora.

Procvičovaná oblast:

Propojování prvků uživatelského rozhraní pomocí signálů.

Kompetence:

Student bude umět 3 způsoby propojení signálů a slotů v knihovně Qt.

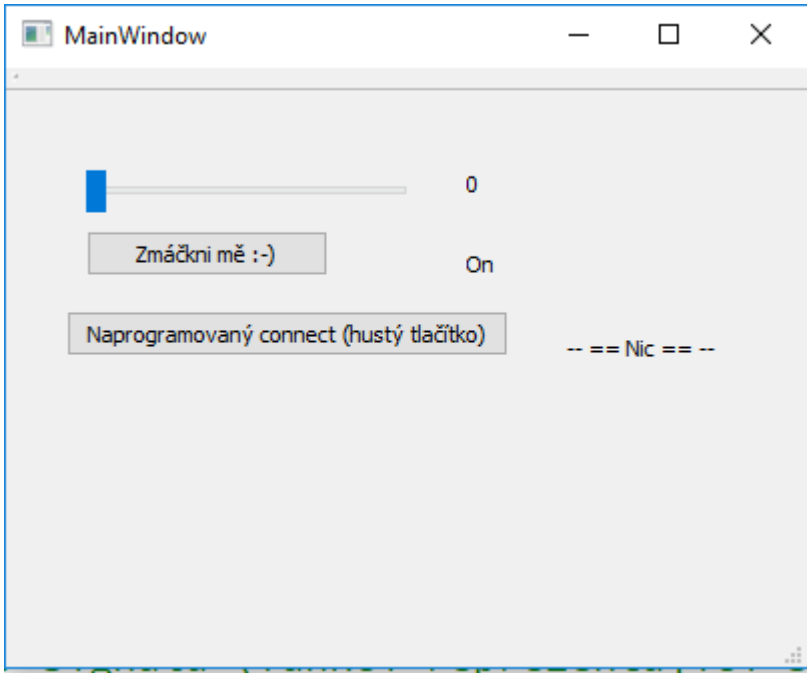
Zadání

- Vytvořte jednoduchou aplikaci, která bude zobrazovat stav posuvníku jako číslo.
- Jiný text bude možné přepínat tlačítkem mezi „On/Off“.
- Zkuste použít pouze funkci connect.

Zkušení žáci

Zkušenější žáci budou pracovat samostatně a používat jiné widgety a hledat možnosti jejich propojování.

Vizualizace



Obrázek 4 Náhled na aplikaci

Ověření

Lekce je jednoduchá, studenti s ní nemají problém, ovšem je nutné dát pozor na správné chápání pojmů a připravit si názorná a pochopitelná přirovnání.

Lekce 2

Cookie Clicker

Teoreticky vychází ze hry cookie clicker, kterou mohou studenti znát. Při návrhu obsahu lekce je tak možné pracovat s jejich představami a není nutné se držet za každou cenu obsahu lekce.

Časová dotace, příprava:

30 minut, naprogramovat celé řešení.

Trvání

Záleží na třídě a na formě spolupráce (nechá-li lektor pracovat studenty) **45 minut – 90 minut.**

Vstupní požadavky na žáka:

Ovládá paradigma objektově orientovaného programování, má ponětí o signálech a slotech v Qt.

Způsob výuky:

Lze zvolit libovolnou variantu. Samostatná práce. Frontální výuka se samostatnou prací podpořená lektorem, nebo frontální výuka s diskusí a společnou prací.

Procvičovaná oblast:

Práce se signály/sloty, platnost proměnné (proměnná třídy), kreativita.

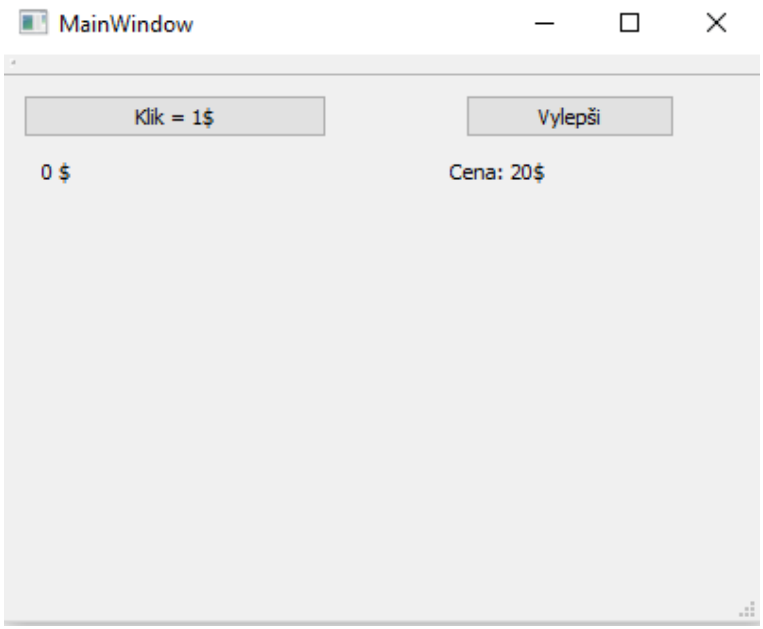
Kompetence:

Student po absolvování lekce bude schopen pracovat s tlačítky, počítat proměnné, dosazovat text.

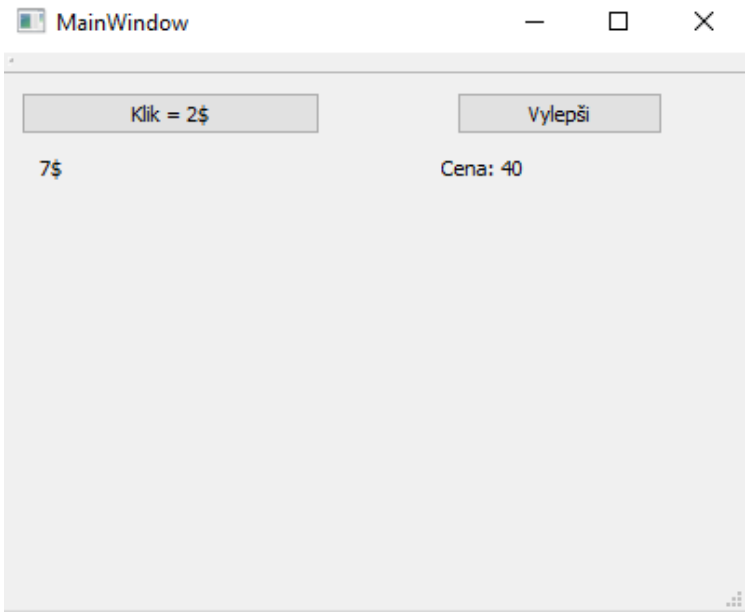
Zadání

- Vytvořte počítadlo klepnutí na tlačítko. Počet vypisujte se značkou dolary ($).
- Vytvořte druhé tlačítko, které při klepnutí zkontroluje, zdali má hráč dost peněz a případně zvýší efektivnost klepnutí.
- Vymyslete způsob výpočtu, jak zdražovat vylepšování a jak zvyšovat výkon klepání.
- Navrhněte další možnosti.

Vizualizace



Obrázek 5 Základní okno



Obrázek 6 Po vylepšení

Ověření

Je důležité studenty navést na práci s proměnnými hlavní třídy, pak už bývá úloha bezproblémová a zvládnutelná v žádaném čase. Pokročilejší třídy mohou začít vymýšlet, jak

by šla aplikace zlepšit. Je vhodné takové odkázat například na získávání peněz za čas, což lze implementovat přes třídu QTimer (viz níže).

7.4.2 Práce s textem a textové widgety v Qt

Qt pro práci s řetězci poskytuje vlastní třídu QString[49]. Na tuto třídu lze implicitně převádět std::string a char *. Pro převádění čísel lze použít statickou metodu `number`[50]. Převádění na číslo jde pomocí metody `.toInt()`, `.toFloat()` nebo `.toDouble()`. Řetězce lze spojovat pomocí operátoru "+".

Třída podporuje běžné operace, jako je zjištění délky (`.length()`), práce na indexech `slovo[3]`, získávání části řetězce, zleva, zprava, uprostřed, atd.

Textové widgety

Základní výstupní widget je QLabel[51]. Zobrazuje jednoduchý textový popisek a podporuje formátování pomocí HTML. Pro nastavení textu lze použít metodu `QLabel::setText(const QString &text)`, pro získání hodnoty v labelu použijeme getter[52] metodu `QString QLabel::text() const`. Stejně se pracuje i s jednořádkovým vstupem – QLineEdit[53].

Víceřádkové vstupy / výstupy jsou implementované ve třídách QTextEdit a v od ní odvozené třídě QTextBrowser. Obě třídy podporují formátování pomocí HTML a lze na základě nich vytvořit textový editor. Metody pro dosazení obsahu jsou `setPlainText(const QString &text)`, která vkládá text "tak jak je", `setHTML(const QString &text)` dosazuje interpretovaný HTML kód a `setText(const QString &text)` se snaží formát odhadnout. Použitím metody `append(const QString &text)` připojujeme text na konec s novým řádkem, tato metoda jde použít pro simulaci výstupu a pro ukládání dat a jejich snadnou reprezentaci jako zpětnou vazbu začínajícím vývojářům.

---

[49] http://doc.qt.io/qt-5/qstring.html

[50] http://doc.qt.io/qt-5/qstring.html#number

[51] https://doc.qt.io/qt-5/qlabel.html

[52] Qt nepoužívá klíčové slovo get u getter metod. Pouze název hodnoty, kterou chcete kopírovat

[53] https://doc.qt.io/qt-5/qlineedit.html

Všechny textové widgety (a platí to nejen pro ně) mají společnou metodu `clear()`, která vymaže obsah.

Lekce 3

Seznam hostů

Textový výstup je později možné nahradit sofistikovanějším grafickým prvek. Zde by se například hodil QTableWidget, nebo QListWidget.

Časová dotace, příprava:

30 minut

Trvání:

45 – 60 minut

Vstupní požadavky na žáka:

Základní fungování Qt a OOP.

Způsob výuky:

Lze zvolit libovolnou variantu. Samostatná práce. Frontální výuka se samostatnou prací podpořená lektorem, nebo frontální výuka s diskusí a společnou prací.

Zkušení žáci:

Zkušení žáci pracují samostatně.

Procvičovaná oblast:

Práce s komplexnějšími textovými výstupy, práce s řetězci (QString).

Kompetence:

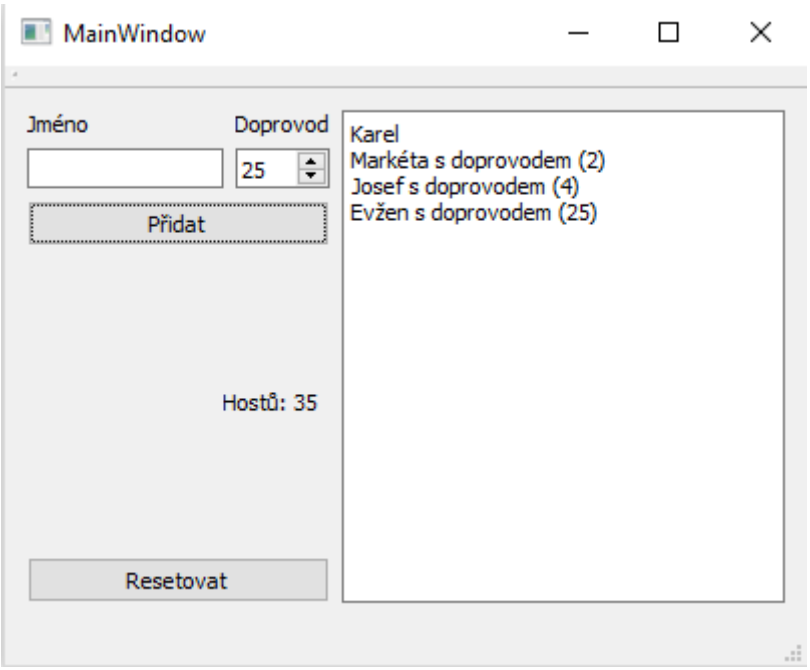
Student dokáže zpracovávat řetězce jako vstup a základní operace (délka, apod.). Také je schopen vypisovat data do víceřádkového textového výstupu.

Zadání

- Vytvořte textový vstup (LineEdit), tlačítko, a textový výstup (TextBrowser).
- Při klepnutí na „Přidat“ přidejte do textového výstupu daného jedince. Nevkládejte prázdný řádek, pokud je vstup prázdný.

- Po přidání vymažte obsah vstupu.
- Přidejte vstup pro přidání doprovodu (počet). Do pole, pokud má jedinec doprovod, připište počet doprovázejících.
- Počet hostů, kolik jich vlastně má přijít, počítejte.
- Přidejte tlačítko „Reset“, které vymaže veškerý obsah.

Vizualizace



Obrázek 7 Náhled na aplikaci

Ověření

Pokud si studenti pamatují princip proměnné dané třídy, lze snadno implementovat počítání hostů. Tato lekce byla problematická pouze při hledání odpovídajícího způsobu dosazování dat do QTextBrowseru

7.4.3 Dialogy v Qt

Hlavní okno může zobrazovat podřízené dialogy. Ke tvorbě takových dialogů se v Qt používá třída QDialog[54]. Vlastní dialogy se vytváří odvozením následníka od této třídy. Pokud chceme vložit dialog do projektu, je možné ho v dialogu *Nový soubor nebo projekt* »

[54] http://doc.qt.io/qt-5/qdialog.html

Qt » *Třída formuláře Qt Designer*. Dialog vytvořený touto cestou vytvoří hlavičkový (.h), zdrojový (.cpp) a soubor s návrhem UI (.ui, upravitelný v návrháři).

Ve výchozím stavu a při vybraném rozvržení je zde widget ButtonBox napojený na `accepted` a `rejected`, což jsou metody pro potvrzení, nebo odmítnutí dialogu. Obě metody dialog skryjí a nastaví mu kód výsledku.

Do dialogu lze vkládat vlastní prvky přímo v návrháři. Data v nich uložená je nejlepší zpřístupnit pomocí getter[55] metod. Pokud je nutné definovat nějaká data dopředu, je to vhodné udělat v konstruktoru, nebo definovat setter[56] metodu nad objektem dané třídy.

Dialogy lze volat jako nemodální pomocí metody `show`, nebo jako modální pomocí metody `exec`. Modální dialog je dialog blokující, což znamená, že uživatel se musí nejprve zpracovat dialog a do té doby nemůže dělat nic jiného.

Již definované dialogy

Od třídy QDialog jsou odvozené některé dialogy[57], které lze v kódu snadno použít. Tyto třídy velmi často definují statické funkce pro jeden specifický účel.

Statické funkce ve třídě `QMessageBox` jsou vhodné pro zobrazení nějaké informace, například informace o zaplněném disku, neplatnosti přihlašovacích údajů apod. Tyto statické metody vrací informaci o tom, které tlačítko bylo zmáčknuto, což je konstanta ze seznamu konstant (enum) `QMessageBox::StandardButton`[58]. Tyto konstanty jsou binární čísla a jejich spojením operátorem `|` [59] lze definovat jiná tlačítka.

Třída `QInputDialog` poskytuje statické funkce pro zadávání některých hodnot (například celého čísla, textu, atd.). Všechny tyto statické funkce lze upravit předáním nepovinných

[55] Getter je termín označující metody, které zpřístupňují obsah členské proměnné některé třídy.

[56] Setter je termín označující metody, které nastavují obsah členské proměnné některé třídy.

[57] http://doc.qt.io/qt-5/qdialog.html - Inherited By

[58] http://doc.qt.io/qt-5/qmessagebox.html#StandardButton-enum

[59] | je binární operátor nebo, označovaný jako binární součet

parametrů, předáním ukazatele na proměnnou typu bool lze na této proměnné zjistit, jestli uživatel neodmítnul dialog potvrdit.

Pokud chceme dialog zobrazit hned při spuštění, je nutné ho umístit do main.cpp do hlavní smyčky těsně před otevření hlavního okna a podmínit ho kladným potvrzením dialogu. V dialogu je nutné přidat dva vstupy a vytvořit pro ně dvě setter metody.

Pokud chceme zajistit validaci dat při potvrzení, je nutné přetížit metodu `accept` a pouze při splnění všech podmínek v přetížené metodě `accept` volat původní metodu z rodičovské třídy `QDialog::accept`. Nekonečným cyklem lze zajistit správné zadané hodnoty před zapnutím programu.

Lekce 4

Přihlašovací okno

Lekci je později možné využít při práci s databázemi, data lze kontrolovat a opravdu implementovat autentifikaci.

Časová dotace, příprava:

30 minut

Trvání

30 - 45 minut

Vstupní požadavky na žáka:

OOP, speciálně pojmy jako „přetěžování“ funkcí, volání původních nepřetížených metod.

Způsob výuky:

Lze zvolit libovolnou variantu. Samostatná práce. Frontální výuka se samostatnou prací podpořená lektorem, nebo frontální výuka s diskusí a společnou prací.

Procvičovaná oblast:

Práce s okny v Qt, dialogy vestavěné i vlastní, zapouzdření, přetěžování funkcí.

Kompetence:

Student chápe systém spouštění Qt programu, návaznost tříd a význam souborů v projektu.

Je schopen zobrazit dialog a zpracovat jeho data, rozumí pojmům přetížení.

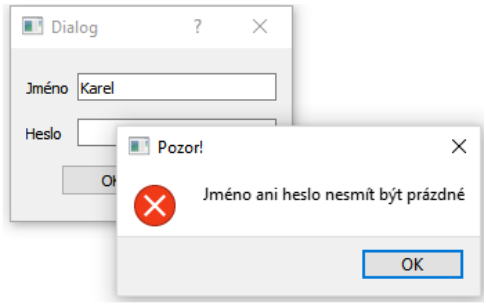
Zadání

- Než se zapne hlavní okno, zobrazte dialog. Pokud uživatel dialog nepotvrdí, nespouštějte hlavní okno.
- V dialogu přidejte vstup pro jméno a heslo. Nechte uživatele pokračovat pouze, pokud vyplní jméno a heslo. Jméno zobrazte v hlavním okně.
- Pokud to nevyplní správně, zobrazte chybovou hlášku, nebo deaktivujte tlačítko.

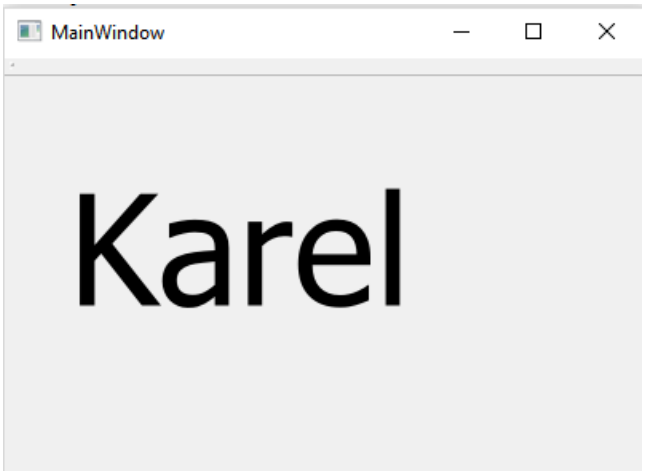
Zkušení žáci:

Zkušení žáci se mohou pokusit implementovat databázi.

Vizualizace



Obrázek 8 Dialog, dialog s chybou.



Obrázek 9 Hlavní okno

Ověření

Někteří žáci byli hotovi rychle, jiným to trvá o mnoho déle, největší problém této lekce je pochopení vytvoření vlastní třídy a přetížení metody `QDialog::accept` pro kontrolu dat. Někteří žáci se snažili aplikaci přizpůsobit (vložení obrázku na pozadí) a simulovat tak přihlašovací okno oblíbené hry (vhodné zařadit jako příklad pro další výuku, je to motivační).

7.4.4 Základní kreslení v Qt

Třídy QGraphicsView, QGraphicsScene

Náhledy na některé grafické prvky poskytuje třída QGraphicsView, která umí zobrazovat různé objekty. Pomocí třídy QGraphicsScene můžeme implementovat dvourozměrné kreslící plátno. Do QGraphicsView dosazujeme objekt QGraphicsScene (vykreslovanou scénu) pomocí metody `setScene(QGraphicsScene *scene)`. Qt ve výchozím stavu se snaží obsah scény zcela vystředit, ale pokud chceme zobrazit pouze určitou část lze definovat v QGraphicsView sceneRect, což je čtverec nastavující výřez pohledu.

QGraphicsScene umožňuje přímo vkládání předmětů a implementuje metody pro základní geometrické tvary (`addRect`, `addEllipse`, `addLine`, `addPolygon`, atd.). Souřadnicový systém je oproti standardnímu kartézskému souřadnicovému systému přetočený podle osy x, což znamená, že záporné y je nahoře. Jde pravděpodobně o důsledek toho, že v počítačích se chápe bod 0, 0 v levém horním rohu a hodnoty na ose y se chápaly jako čísla řádků.

Třída QGraphicsItem

Tato abstraktní třída implementuje jednotlivé objekty, které může uživatel vykreslit, pokud si nevystačí s geometrickými útvary definovanými uvnitř třídy QGraphicsScene nebo je chce seskupit do větších, logických útvarů.

Objekty mají vlastní souřadnice x a y uvnitř QGraphicsScene. Odpovídající metody pro zjištění souřadnic jsou `qreal x()`, `qreal y()` a jejich změnu lze provést metodou `setX(qreal x)`, `setY(qreal y)`.

Pro implementaci následníků této třídy je potřeba implementovat funkce `QRectF boundingRect() const` a metodu `void paint(QPainter *painter, …)`. Metoda `boundingRect` definuje pravoúhelník, ve kterém se objekt vykresluje. Rozměry tohoto pravoúhelníku a pozice jsou relativní vůči absolutní pozici objektu ve scéně, proto typicky má `boudningRect` x a y souřadnici 0 a definuje pouze rozměry. Metoda paint se stará o samostatné vykreslování přes objekt třídy QPainter, který je této metodě předán ukazatelem. Typicky je nutné třídu QPainter vložit. QPainter se pak chová jako scéna, takže je možné vykreslovat úplně stejnými postupy.

Obecně vykreslování v Qt podporuje i Bitmapy, takže je možné vykreslovat grafické soubory. V Qt existuje třída QGraphicsSvgItem, která umožňuje vytvořit QGraphicsItem na základě svg souboru.

Lekce 5

Letadlo

Nebo libovolný obrazec. Pozor na boundingRect(), je nutné, aby definovaný čtverec opravdu obaloval vykreslený objekt. Je také vhodné si nechat studenty vybraný objekt nakreslit a „zakótovat“. Je nutné ovládat princip souřadnicového systému, s tím rozdílem, že y je dole kladné.

Časová dotace, příprava:

30 minut

Trvání:

1 vyučovací hodina

Vstupní požadavky na žáka:

Žáci ovládají kartézskou soustavu množin.

Způsob výuky:

Frontální výuka a později samostatná práce, každý může vytvářet vlastní útvar.

Procvičovaná oblast:

Třídy QGraphicsItem, QGraphicsScene, QGraphicsView.

Kompetence:

Student ovládá základní techniky a postupy při práci s grafikou. Zároveň chápe souřadnicový systém Qt.

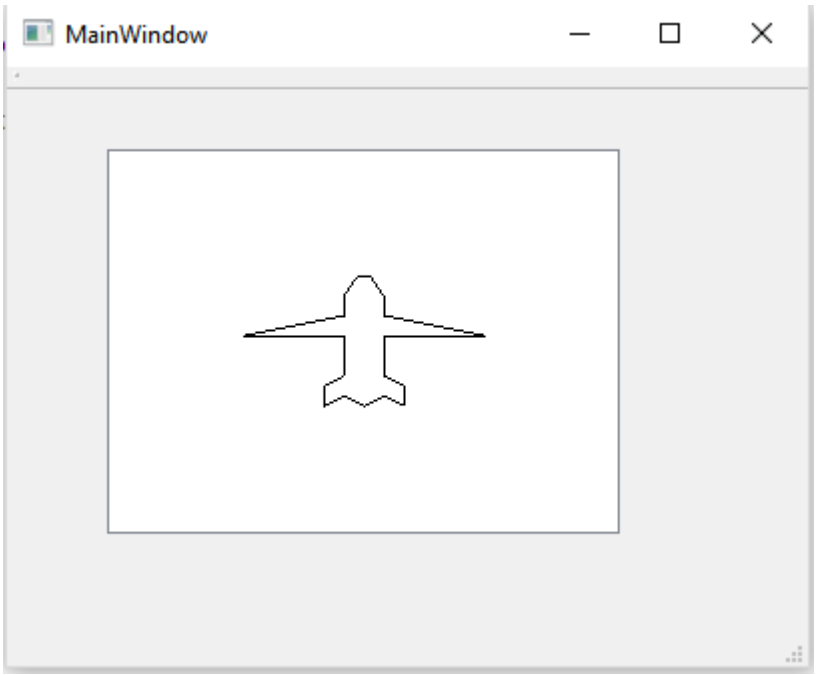
Zadání

- Do programu vložte QGraphicsView a nastavte mu nějakou scénu.
- Vytvořte vlastní třídu, která reprezentuje letadlo a dědí od třídy QGraphicsItem.
- Při vykreslování zaneste všechny body do třídy QPolygon a ten vykreslete.
- Vložte letadlo do scény.

Zkušení žáci:

Zkušení žáci kreslí složitější obrazce

Vizualizace



Obrázek 10 Vykreslené letadlo

Ověření

Je vhodné nechat každého kreslit libovolný útvar, nicméně někteří nechtějí kreslit nic svého, proto je vhodné mít v záloze nějaké jednoduché pro ty, kteří nemají inspiraci. Problém mají

také ti, kterým nejde geometrie. Je vhodné mít čtverečkovaný papír, který lze použít k nakreslení a kótování vlastního obrázku (pro podporu představivosti).

Lekce 6

Hra

Je vhodné vytvořit hru společně s žáky a později je nechat pracovat na samostatných nebo týmových úlohách.

Časová dotace, příprava:

60 minut

Trvání

4 vyučovací hodiny. Rozšířením o domácí projekty může trvání být prodlouženo až na dvojnásobek.

Vstupní požadavky na žáka:

Žáci už pracují s Qt, ovládají signály a sloty včetně manuálního použití funkce „connect“.

Procvičovaná oblast:

Práce s grafikou, časovač, ovládání vstupu z klávesnice, práce s náhodou.

Způsob výuky:

Společná práce na jedné konkrétní hře a později rozdělení samostatných projektů a kontrola formou konzultací a podpory lektora.

Kompetence:

Student ovládá grafické prvky, umí zpracovávat opakované události nebo stisknutí klávesy. Umí využít náhodné jevy.

Zadání

- Vytvořte vlastní třídy „GameView“, která dědí od QGraphicsView a dosaďte ji jako hlavní widget okna. Je vhodné správně nastavit velikosti.
- V GameView vytvořte scénu, která má fixní rozměry.
- Vytvořte objekt hráče, při stisknutí kláves doleva, doprava se hráč hýbe (událost je nutné zachytávat na třídě GameView).

- Hráč nesmí opustit herní plochu.
- Vytvořte třídu kámen, která se objevuje na náhodné pozici nahoře na scéně.
- Vytvořte časovač a slot, který reaguje opakovaně. Při každém spuštění slotu posuňte kámen o trochu níž, jakmile se dostane mimo obrazovku, tak ho přesuňte nahoru.
- Využijte třídu QList a vytvářejte kameny, padne-li číslo menší než nějaká hranice. Takto vytvořené kameny ukládejte do QListu a při každém spuštění časovače je všechny posuňte.
- Pokud se kámen dostane mimo obrazovku, smažte jej ze scény a ze seznamu.
- Vytvořte počítadlo kolizí. Pokud hráč koliduje s jakýmkoliv kamenem, zvyšte počet kolizí a kámen odstraňte ze scény a ze seznamu. Počet kolizí zobrazujte na obrazovce

Zkušení žáci:

Zkušení žáci rovnou pracují na samostatných projektech.

Vizualizace

Obrázek 11 Náhled na hru

Ověření

Lekce je pro studenty velmi atraktivní a jsou velmi aktivní. Je nutná aktivní podpora při jejich projektech.

7.4.5 Zpracování vstupu klávesnice

Všechny třídy odvozené od QWidget (což jsou prakticky všechny prvky uživatelské rozhraní) a některé další implementují metody `keyPressEvent` a `keyReleaseEvent`, metoda jsou virtuální, takže ji lze přetížit a využít k reakci na stisknutí klávesy. K vyvolání funkce dochází pouze v případě, že je objekt zaměřený. Pokud chceme používat zaměřování pomocí myši pro objekty typu QGraphicsItem uvnitř QGraphicsItem, tak je nutné jim nastavit vlastnost[60] `QGraphicsItem::ItemIsFocusable` na `true`.

Tyto metody předávají ukazatel na objekt třídy QKeyEvent, kde je možné zjišťovat kód klávesy (k tomu slouží metoda `Qt::Key code()`. Kódy kláves jsou definovány ve jmenném prostoru Qt pomocí `enum` Qt::Key.

7.4.6 Časované události

Pro vytváření časovačů lze použít třídu QTimer. Objekty třídy QTimer je vhodné vytvářet vždy v kontextu té třídy, ve které bude použit. Velmi snadno ho tak lze použít uvnitř námi definované třídy QGraphicsView, kterou lze použít jako widget pro zobrazení herní scény (při vytváření vlastní hry). Při vytváření objektu třídy QTimer je také možné předat ukazatel na rodičovský prvek[61]. Pokud je rodičovský prvek nastavený, bude QTimer vyvolávat metodu `timerEvent`, nebo lze manuálně vytvořit propojení přes funkci `connect` na vlastní slot.

Při tvorbě hry je vhodné mít vytvořenou třídu, která bude dědit od třídy QGraphicsView, který bude řídit celou logiku hry. Objekt je nutné vytvořit dynamicky ve třídě hlavního okna a dosadit pomocí `setCentralWidget(QWidget *widget)`. Pokud je centrální widget nastaven přímo na jeden konkrétní widget nelze využívat Qt Designer pro dosazování widgetů do centrální oblasti. Ovládací prvky je možné vkládat i nadále do QDockWidgetu (dokovací okno).

[60] Qt používá termín "flag", od toho je odvozená metoda setFlag

[61] Qt používá princip rodičovských prvků pro vytvoření systému logických posloupností, které může využít při správě paměti.

QGraphicsTextItem je třída, která vykresluje text do scény. Pomocí tohoto prvku lze zobrazit skóre, čas apod. Při nastavování lze využít i metodu `setHTML`, která může použít i text s HTML značkami (tučné písmo, kurzíva, aj.)

7.4.7 Seznamy a tabulky založené na prvcích

Qt implementuje i seznamy (QListWidget), nebo tabulky (QTableWidget). Tyto widgety lze použít k zobrazení většího množství dat, například výstup databáze, výstup hledání atd. Tyto widgety jsou "item-based" (založené na prvcích).

Třída QListWidget

QListWidget zobrazuje seznam jako řádky pod sebou. Řádky je možné označovat a nastavit i režim označení (jeden, či více prvků). Jednotlivé záznamy (objekty třídy QListWidgetItem) mohou obsahovat ikony, nebo pouze text. Vkládání záznamů se dělá metodou `addItem`, která jako parametr bere text (vytvoří běžný čistě textový záznam), nebo ukazatel na objekt třídy QListWidgetItem, který může mít nastavené nějaké další informace. Prvky jsou jednoznačně identifikovány ukazatelem, nebo pozicí (číslem řádku) uvnitř QListWidget.

Funkce `selectedItems` vrací QList ukazatelů na všechny vybrané objekty (což může být i QList s jedním objektem). Nastavení vlastního skrytého obsahu lze provést pomocí metody `setData`, která umožňuje nastavit určitá data prvkům, jako `int role`, nastavíme `Qt::UserRole`, která umožní nastavit libovolná data. Následující příklad vygeneruje 10 položek očíslovaných od 0 do 9, které mají svoje číslo uložené i uvnitř. Data se ukládají jako QVariant, což je kontejner pro různé datové typy a je nutné ho pro používání přetypovat zpět na požadovaný datový typ. (např. `toInt()`, `toString()`)

```
for(int i=0; i<10; ++i)
{
  QListWidgetItem *item;
  item = new QListWidgetItem("Položka - " + QString::number(i));
  item->setData(Qt::UserRole, i);
  ui->listWidget->addItem(item);
}
```

Mazání prvků funguje na úrovni čísel řádků. Je potřeba zjistit číslo aktuálního prvku `currentRow()`, nebo všech vybraných prvků `selectedRows()`. Druhá metoda bere

pouze aktivně vybrané prvky a vrací jejich ukazatele v QListu. Ve výchozím stavu není možné vybrat více prvků najednou.

Odebírání prvků z tohoto widgetu se dělá pomocí metody "`takeItem`", která odebere jeden řádek (předává se číslo řádku) a vrací ukazatel na prvek, který je na daném řádku. Takový prvek je pak nutné pomocí `delete` smazat. Pokud je přístupný prvek (QListWidgetItem) přes ukazatel, získává se číslo řádku metodou `QListWidget::row(QListWidgetItem *item)`.

Lekce 7

Nákupní seznam

Nabízí se možnost rozšířit aplikaci o ukládání do databáze, nebo napojení na veřejné API a sdílet nákupní seznam s webovou aplikací.

Časová dotace, příprava:

45 minut

Trvání:

2 vyučovací hodiny

Vstupní požadavky na žáka:

Chápání ukazatelů a správy paměti v C++, znalost základních konceptů Qt.

Způsob výuky:

Lze zvolit libovolnou variantu. Samostatná práce. Frontální výuka se samostatnou prací podpořená lektorem, nebo frontální výuka s diskusí a společnou prací.

Procvičovaná oblast:

Práce s QListWidget, správa prvků.

Způsob výuky:

Lze zvolit libovolnou variantu. Samostatná práce. Frontální výuka se samostatnou prací podpořená lektorem, nebo frontální výuka s diskusí a společnou prací.

Kompetence:

Studenti si osvojí základy komplexnějších widgetů, jako je např. QListWidget a operace s jejich prvky.

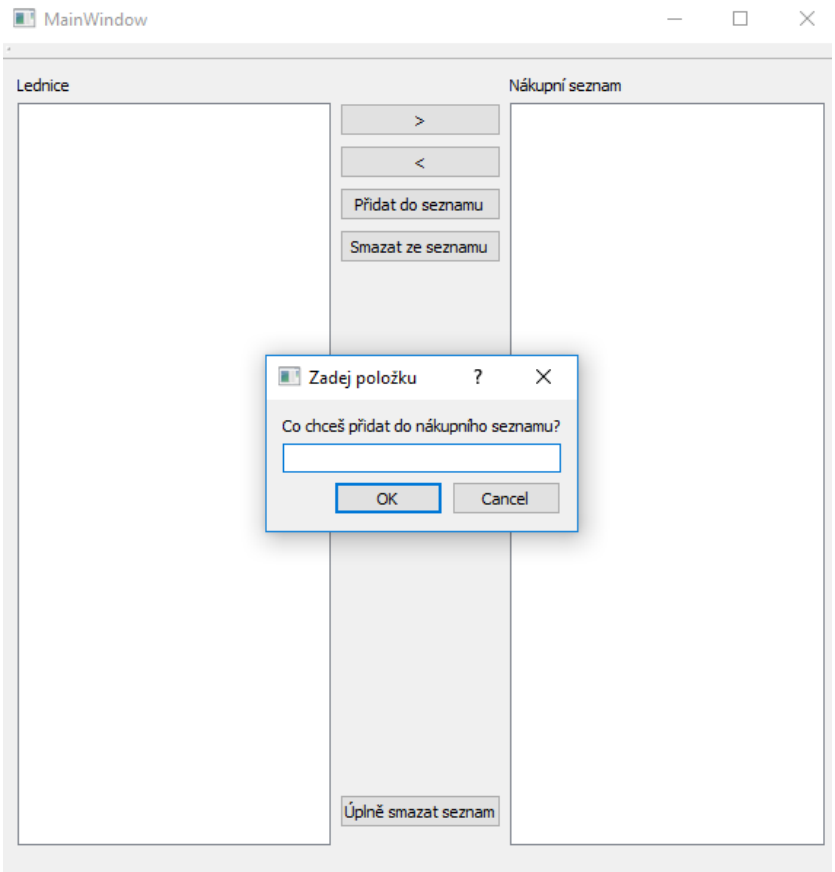
Zkušení žáci:

Zkušení žáci pracují samostatně, případně pracují na svých projektech.

Zadání

- Vytvořte aplikaci. Bude potřeba mít dva seznamy, dvě tlačítka pro přesouvání doprava i doleva. Tlačítko pro přidání do seznamu.
- Vytvořte tlačítko, které odebere všechny prvky ze seznamu.
- Vytvořte tlačítko, které vymaže vybrané prvky ze seznamu.

Vizualizace



Obrázek 12 Náhled na aplikaci

Ověření

Při vývoji je nutné se zaměřit na vysvětlení správy paměti a přesouvání prvků mezi oběma seznamy. Správně zvládnutý výklad zjednodušuje výklad složitějších widgetů, je nutné se na toto téma zaměřit.

Třída QTableWidget

TableWidget zobrazuje tabulková data, podobně jako např. Excel. Data jsou uspořádána ve dvourozměrné matici a jsou indexována číslem řádku a číslem sloupce.

Při vytvoření a vložení widgetu se definuje počet sloupců této tabulky metodou `setColumnCount`. Sloupce jsou očíslované, nastavení popisků lze provést metodou `setHorizontalHeaderLabels`, která definuje popisky jednotlivých sloupců. Pokud nemají některé sloupce být viditelné, lze je skrývat pomocí metody `setColumnHidden`.

Počet řádků lze definovat na začátku (`setRowCount`), nebo je lze vkládat v kódu pomocí funkce `insertRow`. (dodatečně lze i vložit sloupce pomocí `insertColumn`).

Jednotlivé buňky jsou prvky třídy QTableWidgetItem a vkládání dat lze dělat přes metodu `setItem`, která na zadané souřadnice (řádek, sloupec) vloží právě QTableWidgetItem. Lze odebírat řádky, sloupce, vymazání jednotlivých buněk lze provést nastavením jejich obsahu na prázdný řetězec. Pro výběr prvků lze použít funkce `itemAt`, která vrací ukazatel na QTableWidgetItem.

```
ui->tableWidget->setItem(0, 1, new QTableWidgetItem("[0, 1]"));
```

QTableWidget umí vícenásobný výběr, vybírat prvky pouze po řádcích, sloupcích apod. Vybrané prvky jsou přístupné přes funkci `selectedItems()`. Prvky tabulky obsahují (na rozdíl od položek QListWidget) i informaci o řádku (metoda `row`) a sloupci (metoda `column`).

Lekce 8

Teplota

Aplikace by mohla stahovat data z některé veřejné API.

Časová dotace, příprava:

45 minut

Trvání

2 vyučovací hodiny

Vstupní požadavky na žáka:

Práce s Qt, OOP, znalost SQL.

Procvičovaná oblast:

Práce s databází, model/view, grafika.

Způsob výuky:

Frontální výuka, uvést podobné příklady a případně formou samostatné práce (s podporou lektora), nebo společně vypracovat příklad.

Kompetence:

Student se naučí pracovat s databází a základními databázovými dotazy. Vyzkouší si použití model/view konceptu v Qt a vykresluje data do QGraphicsView.

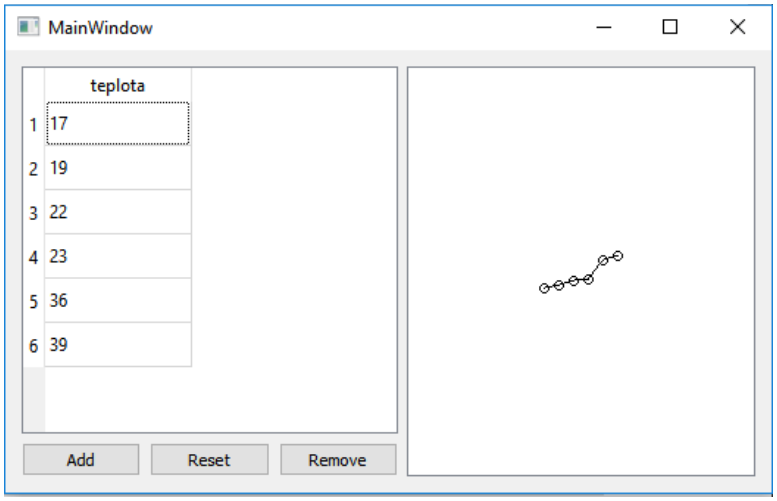
Zkušení žáci:

Zkušení žáci mohou implementovat vlastní projekt, nebo lze vytvořit komplexnější databázi s navazujícími tabulkami.

Zadání

- Vytvořte tabulku v relační databázi, každý prvek má ID (primární klíč, auto increment)a teplotu (int).
- Tuto tabulku napojte na QTableView přes vytvoření QSqlTableModel a zobrazte. Omezte vybírání prvků na jeden a po řádcích.
- Přidejte tlačítko „plus“, které zobrazí dialog pro zadání teploty a uloží ji do databáze.
- Přidejte tlačítko „reset“, které vymaže všechna data (i z QGraphicsView).
- Zobrazte graf do QGraphicsView.
- Přidejte tlačítko „minus“, které smaže jeden vybraný záznam.

Vizualizace



Obrázek 13 Náhled na aplikaci

Ověření

Technická stránka věci, použití modelu, není natolik problematická, jako vysvětlení principu grafu, kde je nutné pamatovat si předchozí hodnotu pro vykreslování.

7.4.8 Model-based widgety

Velké množství widgetů obsahuje tzv. model. Model vychází z konceptu Model/View, který odděluje data od prezentační vrstvy. Na rozdíl od předcházejících příkladů, kde pracujeme s jednotlivými prvky, tento přístup definuje pohled (jak vypadá zobrazovací widget) a do něj dosazujeme data, která jsou nějakým způsobem koherentní (například jsou uložená v jedné tabulce databáze). Modely lze definovat vlastní, lze vytvářet objekty a do nich dosazovat data, ale jedno z nejzajímavějších použití je přímá integrace databázových pohledů.

Připojení databáze

Pro používání databázových pohledů je nutné aktivovat část Qt, která má na starost práci s databázemi. Tuto vlastnost nastavíme v projektovém souboru doplnění `sql` za seznam komponent.

```
QT += core gui sql
```

Qt podporuje různé databázové systémy a různé způsoby prací s databázemi. První nutnou činností je databázi připojit. Připojení databáze vytváří spojení, které je uloženo jako statická

proměnná uvnitř třídy QSqlDatabase, což znamená, že spojení s databází není nutné předávat jako proměnnou, ale k předávání dochází automaticky. Přesto je vhodné databázi nastavit jako členskou proměnnou některé třídy (například třídy hlavního okna).

Databázové spojení se vytváří pomocí statické metody `QSqlDatabase::addDatabase("QSQLITE")`. Řetězec předávaný jako parametr určuje typ ovladače pro připojení (a de facto i druh databáze). Definují se přístupové údaje (jméno, heslo, název databáze) a pomocí metody `open` (metoda vrací true/false, podle toho, pokud se spojení do databáze podaří otevřít).

QSqlTableModel

QSqlTableModel je třída, která propojuje data z databázové tabulky s QTableView. Pokud je připojená databáze, určuje se tabulka metodou `setTable`, která jako parametr bere název tabulky. Do QTableView lze model dosadit metodou `QTableView::setModel`, takto dosazený model nemá načtená ještě data. Načtení dat z databáze a jejich případné obnovení se provede metoudou `select()`. Data je možné filtrovat pomocí metody setFilter, kde se používá stejný zápis jako v SQL podmínce WHERE, viz následující příklad. Filtr se aplikuje při aktualizaci dat metodou `update()`.

```
model->setFilter("name='Karel'");
```

Další způsoby práce s databázemi

Qt nabízí další možnosti práce s databázemi, například třída QSqlQuery, která umožňuje používání standardních SQL dotazů. Třída podporuje připravování metodou `prepare` a dosazování hodnot metodou `bindValue`.

7.4.9 Síťové aplikace

Síťovou komunikaci zprostředkovává Qt pomocí tříd QTcpSocket, nebo QUdpSocket, serverovou část implementují třídy QTcpServer, QUdpServer. Síťové aplikace mohou rozšířit možnosti komunikace, síťové hry apod.

Práce s běžnými internetovými zdroji je možná přes třídu QNetworkAccessManager, která umožňuje běžné GET/POST požadavky jako objekty třídy QNetworkRequest. Tento typ lze použít pro napojení se na JSON/REST API. Zde je vhodné zmínit, že Qt podporuje i JSON a XML parsování.

7.4.10 Další komponenty

Další zaměření může směřovat na práci s obrázky (QPixmap), soubory (QFile) atd. Qt nabízí velké množství možností a vzhledem k velké komunitní základně tak jde rozšířit základní funkčnost ještě o existující třídy a celé knihovny (NoSQL databáze a pohledy atd.)

8 Závěr

8.1 Teoretická část

V práci jsme provedli mapování školních vzdělávacích programů s ohledem na výuku vývoje aplikací s grafickým uživatelským rozhraním. Školní vzdělávací programy se velmi liší, část škol v nich tvorbu aplikací s GUI vůbec nezmiňuje, jiné se liší v použitých technologiích.

Bylo provedeno i mapování trhu práce a tzv. popularity daných technologií s cílem vybrat širší množinu jazyků pro následnou analýzu. Jako nejpoužívanější (popularita) a nejžádanější (trh práce) se ukázaly být jazyky Java, C++, C, Python, C#, PHP.

Výše uvedená skupina programovacích jazyků byla popsána a analyzována podle předem stanovených kritérií. Z této skupiny byly pro podrobnější analýzu vybrány programovací jazyky C++ a Java. Z těchto dvou pak byl zvolen jazyk C++, které nabízí širší množství vlastností, například v OOP.

Po výběru programovacího jazyka bylo obdobně postupováno i v otázce výběru grafického frameworku. U každého z frameworků (využitelných ve spojení s jazykem C++) byly popsány specifické vlastnosti a jeho rozšíření v praxi. Následně byl vybrán framework Qt jako multiplatformní framework s velkou uživatelskou základnou a dostupným vývojovým softwarem.

V poslední části práce jsou pak popsány nástroje používané při tvorbě Qt aplikace (např. vývojové prostředí). Dále jsou rozebrány jednotlivé součásti knihovny a naznačen způsob

práce s nimi. Komponenty mají připravené ukázkové lekce, kde je možné funkčnost snadno otestovat.

8.2 Ověření použitelnosti lekcí

Vytvořené lekce byly ověřeny na několika nezávislých vyučovaných skupinách. Skupiny se lišily věkem, zkušenostmi i zaměřením školy.

3. ročník, Střední průmyslová škola, Mladá Boleslav

Studenti znají základy C++ včetně objektově orientovaného programování. Studenti byli seznámeni se základy Qt. Úvodní lekce (1. až 5. lekce) byly víceméně bezproblémové, při kombinaci postupů ve vyšších lekcích byla stále větší potřeba dopomoci lektora. Během výuky bylo také nutné opakovat předchozí látku. Časová náročnost odpovídala časům stanoveným v jednotlivých lekcích.

4. ročník, Střední průmyslová škola, Mladá Boleslav

Studenti již měli pokročilou znalost C++ a konceptů při programování používaných. Základní lekce (1. až 6. lekce) pro ně byli velmi jednoduché a zvládli je za polovinu času. Další lekce vycházeli na běžnou dobu, kdy studenti z větší části pracovali samostatně, některé složitější věci bylo nutné vysvětlit (koncept model → view, apod.).

4. ročník, Gymnázium dr. Josefa Pekaře, Mladá Boleslav

Další sledovaná skupina byla studentů volitelného semináře programování na Gymnáziu dr. Josefa Pekaře. Studenti studují všeobecně vzdělávací obor, ale volí si zaměření formou předmětů. Tito studenti programují v programovacím jazyce Pascal, který s C++ sdílí některé základní teoretické možnosti (pointer), nicméně syntaxe se liší. Studentům bylo potřeba vždy vyložit danou konstrukci v C++, pak byly schopni nápodobou a různými variacemi lekce zpracovávat (lekce 1-4). Pozdější lekce kvůli absenci znalosti objektově orientovaného programování bylo nutné předělat, nebo vypustit. Časová složitost uvedená v lekcích neodpovídala skutečnosti.

9 Seznam použitých informačních zdrojů

1. Ministerstvo školství, mládeže a tělovýchovy. Rámcový vzdělávací program pro obor vzdělávání. [Online] http://zpd.nuov.cz/RVP/ML/RVP%201820M01%20Informacni%20technologie.pdf.

2. Smíchovská střední průmyslová škola. Profil absolventa oboru IT. [Online] [Citace: 4. 8 2016.] http://www.ssps.cz/candidate/read/119.

3. SPŠ a VOŠ Písek. Učební osnovy pro ŠVP: č. j. SPŠ/1380/2014. [Online] [Citace: 8. 4 2018.] http://www.sps-pi.cz/wp-content/uploads/2016/12/%C5%A0VP_IT_U%C4%8Debn%C3%ADOsnova_Od_2014-2015_Z02_Web.pdf.

4. Střední škola informatiky a cestovního ruchu SČMSD Humpolec, s.r.o. Školní vzdělávací program - Informační technologie. [Online] [Citace: 8. 4 2018.] http://www.stredniskola.com/svp/it.pdf.

5. Programování. *Soukromá střední škola výpočetní techniky.* [Online] [Citace: 15. 4 2018.] https://www.sssvt.cz/informace/predmety_programovani/.

6. Zieleniecová, Pavla. Pedagogika II. *Pedagogika v učitelském studiu M - F - I - Dg.* [Online] [Citace: 08. 4 2018.] https://kdf.mff.cuni.cz/vyuka/pedagogika/materialy/2015%20LS/20150311%20Pedagogika%20II%20-%206%20prednaska%20LS%202014-15.pdf.

7. Stack Exchange Inc. Trends. *Stack Overflow Insights.* [Online] [Citace: 8. 4 2018.]

8. Robinson, David. Introducing Stack Overflow Trends . *Stack Overflow blog.* [Online] 9. 5 2017. [Citace: 8. 4 2018.] https://stackoverflow.blog/2017/05/09/introducing-stack-overflow-trends/.

9. TIOBE. TIOBE Index for April 2018. [Online] [Citace: 8. 4 2018.] https://www.tiobe.com/tiobe-index/.

10. Hudgens, Jordan. Developer Learning Curve – Why Learning How to Code Takes So Long? *Crondose.com.* [Online] 1. 9 2016. [Citace: 8. 4 2018.]

11. Beneš, Nikola. Programovací jazyky. *IB111 Úvod do programování skrze Python.* [Online] 14. 12 2016. [Citace: 04. 08 2018.]

12. Běhálek, Marek. Programovací jazyk C#. [Online] [Citace: 8. 4 2018.]

13. Bořánek, Roman. Microsoft otevře platformu .NET. Poběží i na Linuxu . *Root.cz.* [Online] 12. 11 2014. [Citace: 8. 4 2018.]

14. Kernighan, Brian W. a Ritchie, Dennis M. *The C programming language.* Englewood Cliffs : N.J.:, 1978. ISBN 0-13-110163-3.

15. Stroustrup, Bjarne. The C++ Programming Language. [Online] [Citace: 8. 4 2018.] http://www.stroustrup.com/C++.html.

16. Herout, Pavel. *Učebnice jazyka Java.* České Budějovice : Kopp, 2010. ISBN: 978-80-7232-398-2.

17. *WxWidgets.* [Online] [Citace: 15. 4 2018.] https://www.wxwidgets.org/.

18. Přispěvatelé. Qt History. *Qt Wiki.* [Online] [Citace: 08. 04 2018.]

19. Building Firefox for Windows. *MDN web docs.* [Online] [Citace: 15. 4 2018.] Building Firefox On Windows. Stack Overflow blog [online]. [cit. 2018-04-15]. Dostupné z: https://developer.mozilla.org/en-US/docs/Mozilla/Developer_guide/Build_Instructions/Windows_Prerequisites.

20. Ježek, David. Google začíná i Chrome pro Windows sestavovat pomocí LLVM/Clang . *Root.cz.* [Online] 6. 3 2018. [Citace: 15. 4 2018.] https://www.root.cz/zpravicky/google-zacina-i-chrome-pro-windows-sestavovat-pomoci-llvm-clang/.

21. Lopatkin, Ilya. Non-ASCII path names with qmake on Windows. *Qt bug reports.* [Online] [Citace: 15. 4 2018.] https://bugreports.qt.io/browse/QTBUG-6080.

22. The Qt Company Ltd. Qt Documentation. [Online] [Citace: 15. 4 2018.] http://doc.qt.io/.

23. ONeal, Billy. Why aren't more desktop apps written with Qt? [closed]. *Stack Exchange.* [Online] [Citace: 15. 4 2018.] https://softwareengineering.stackexchange.com/questions/88685/why-arent-more-desktop-apps-written-with-qt.

24. Phan, Thomas X., Montanari, Rebecca a Zerfos, Petros. *Mobile computing, applications and services, first international ICST conference, MobiCASE 2009 San Diego, CA, USA, October 26-29, 2009, revised selected papers.* New York : Springer, 2010. ISBN 978-364-2126-062.

25. Lazarevič, Arsen. Práce v IT: Kde jsou největší platy a mzdy? *Měšec.cz.* [Online] 15. 11 2012. [Citace: 04. 08 2018.] https://www.mesec.cz/clanky/prace-v-it-kde-jsou-nejvetsi-platy-a-mzdy/.

26. SPŠ Ostrava. Školní vzdělávací program Informační technologie od šk. roku 2012/2013 pro rok 2017/2018. [Online] [Citace: 8. 4 2018.] http://www.spseiostrava.cz/cs/component/phocadownload/category/33-svp?download=596:svp-it-2012-upr-2017.

27. Soukromá střední škola výpočetní techniky. Specializace programování a databázové systémy. [Online] [Citace: 8. 4 2018.]

10 Seznam příloh

Příloha 1 – Popis lekcí

Příloha 2 – Vypracované projekty