Moderní metody výuky programování na střední škole

Bc. Michal Kostka

Diplomová práce
2017

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2016/2017

ZADÁNÍ DIPLOMOVÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Michal Kostka**
Osobní číslo:	**A15341**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Učitelství informatiky pro střední školy**
Forma studia:	**prezenční**
Téma práce:	**Moderní metody výuky programování na střední škole**
Téma anglicky:	**Modern Programming Tuition in Secondary School Methods**

Zásady pro vypracování:

1. **Seznamte se s problematikou a současnými trendy vytváření výukových materiálů na středních školách.**
2. **Prostudujte si vybraný Rámcový vzdělávací program pro střední školy, přičemž se zaměřte na oblast výuky programování.**
3. **Navrhněte vhodnou metodiku pro výuku programování s využitím moderních technologií jako jsou multimédia, webové aplikace apod.**
4. **Navrhněte vhodný výukový modul do systému Moodle s daným vzdělávacím obsahem.**
5. **Ověřte navržené postupy při výuce na střední škole a výuku vyhodnoťte.**

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **Rámcový vzdělávací program pro obor vzdělání 18-20-M/01 Informační technologie. Národní ústav odborného vzdělávání, Praha, 2007. Dostupné na URL http://zpd.nuov.cz/RVP/ML/RVP%201820M01%20Informacni%20technologie.pdf (leden 2017).**
2. **LEPIL, O. Teorie a praxe tvorby výukových materiálů. Univerzita Palackého, Olomouc, 2010. ISBN 978-80-244-2489-7. Dostupné na URL http://zvyp.upol.cz/publikace/lepil.pdf (září 2012).**
3. **ČANDÍK, M. a Š. CHUDÝ. Didaktika informatiky. Zlín: Univerzita Tomáše Bati ve Zlíně, 2005, 133 s. Učební texty vysokých škol. ISBN 8073182858.**
4. **TURECKIOVÁ, M. a J. VETEŠKA. Kompetence ve vzdělávání. Grada Publishing, Praha, 2008.**
5. **DRLÍK, M. Moodle: kompletní průvodce tvorbou a správou elektronických kurzů. Brno: Computer Press, 2013, 344 s. ISBN 978-80-251-3759-8.**

Vedoucí diplomové práce: **doc. Ing. Jiří Vojtěšek, Ph.D.**
Ústav řízení procesů

Datum zadání diplomové práce: **3. února 2017**

Termín odevzdání diplomové práce: **16. května 2017**

Ve Zlíně dne 3. února 2017

doc. Mgr. Milan Adámek, Ph.D.
děkan

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

prof. Mgr. Roman Jašek, Ph.D.
ředitel ústavu

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 10. 5. 2017

.....
podpis diplomanta

ABSTRAKT

Diplomová práce je zaměřena na výuku programování na střední škole. Práce se zabývá návrhem volitelného předmětu zaměřeného na programování, metodami výuky programování pro (nejen) tento předmět a tvorbou materiálů sloužících k výuce programování. Výstupem práce je také kurz v systému Moodle se vzdělávacím obsahem pro tento předmět. Předpokládá se využití výstupů práce při výuce oboru IT na Obchodní akademii Kroměříž.

Klíčová slova: výuka, programování, střední škola, metody výuky, kurikulum, Moodle, učební materiály.

ABSTRACT

This thesis is focused on programming tuition at secondary school. The thesis deals with the design of an optional school subject focused on programming, programming tuition methods for (not only) this subject and the creation of materials used for teaching of programming. The output of the thesis is also a course in Moodle with educational content for this subject. It is assumed to use the outputs of the thesis for IT programme tuition at the Business Academy Kroměříž.

Keywords: education, programming, high school, teaching methods, curriculum, Moodle, learning materials.

Rád bych poděkoval doc. Ing. Jiřímu Vojtěškovi, Ph.D. za odborné vedení a důležité rady pro vypracování této práce. Dále bych rád poděkoval řediteli Obchodní akademie Kroměříž PhDr. Mojmíru Šemnickému, MBA a pracovníkům této školy, především Ing. Soni Kartousové, Jiřímu Kouřilovi a Mgr. Tomáši Valachovi, za poskytnutí školních dokumentů, podporu pro realizaci praktické části ve školním systému Moodle a umožnění provedení ověřování. V neposlední řadě bych chtěl poděkovat svým přátelům, kteří mi poskytli potřebné odreagování při tvorbě práce, přestože mi ho občas poskytovali až moc.

Prohlašuji, že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Moderní informační technologie jsou stále důležitější součástí dnešní společnosti. Roste počet lidí, kteří je používají denně. Objevují se pojmy jako „internet věcí“ (Internet of Things), což znamená napojení běžných strojů a přístrojů na internet – televize, rádio, ledničky, pračky, dopravní prostředky a další. Již dnes je součástí většiny z těchto nově vyrobených přístrojů programovatelný mikrokontrolér nebo počítač. Hovoří se také o průmyslu 4.0, tedy o další vlně nahrazování lidských pracovních sil stroji, které budou muset být pravděpodobně také naprogramovány.

Zdá se tedy, že úloha znalosti programování ve společnosti bude růst. Mluví se o nedostatku odborníků v technických oborech, a to i programátorů. Částečně na to reagují školy, které otevírají obory tohoto nebo blízkého zaměření.

Na Obchodní akademii Kroměříž, škole zřizované Zlínským krajem, je od 1. 9. 2014 vyučován obor Informační technologie. Pevnou součástí tohoto oboru je výuka programování. Cílem této práce je připravit materiály a metodiku výuky pro nový volitelný předmět zaměřený na rozšíření kompetencí v oblasti programování.

Při práci bych rád navázal na předchozí práci, kdy jsem vytvářel Sbírku příkladů z programování. [1] Dále bych rád využil zkušenosti, které jsem získal praxí, kdy jsem více než tři roky pracoval jako programátor v malé soukromé firmě.

Teoretická část je rozdělena na tři kapitoly. První kapitola práce se teoreticky zabývá výukou a metodami výuky. V kapitole 2 jsou shrnuty dokumenty, které definují požadavky na výuku programování. Třetí kapitola řeší teorii tvorby výukových materiálů.

V praktické části kapitola 4 popisuje návrh nového volitelného předmětu zaměřeného na programování pro Obchodní akademii Kroměříž. Kapitola 5 navrhuje výukové postupy pro výuku programování. Kapitola 6 popisuje využití systému Moodle. Poslední, sedmá kapitola se zabývá ověřováním navržených výukových postupů.

I. TEORETICKÁ ČÁST

1 VÝUKA A METODY VÝUKY

Tato kapitola se zabývá především definicí pojmů jako výuka a metody výuky a dále rozlišením „tradičních“ a „moderních“ metod výuky. V kapitole 1.1 je rozebrán pojem výuka, jeho souvislost s pojmem vyučování a její nejdůležitější komponenty. V části 1.2 jsou definovány metody výuky a popsáno jejich rozdělení na tradiční a moderní.

1.1 Výuka

Výuka je specifický druh lidské činnosti, který spočívá v součinnosti učitele a žáků (žáka), který směřuje k určitým cílům. Pojem výuka je někdy zaměňován s pojmem vyučování, který se ovšem vztahuje výhradně k činnosti učitele. Nejdůležitější komponenty výuky tvoří cíle vyučovacího procesu, obsah (učivo), součinnost učitele a žáka a didaktické prostředky. [2]

Didaktické prostředky členíme na materiální a nemateriální. Materiálními didaktickými prostředky rozumíme různé učebnice, didaktickou techniku, školní budovy, hmotné pomůcky apod. Nemateriálními didaktickými prostředky rozumíme organizační formy a metody výuky. [3]

Za povšimnutí stojí fakt, že materiální didaktické prostředky v dnešní době nemají vždy materiální povahu, tzn. že si na ně můžeme sáhnout. Učebnice a různé další učební pomůcky mohou mít v dnešní době virtuální, elektronickou podobu. Přesto jsou považovány za materiální prostředek.

1.2 Metoda výuky

Metoda výuky je jednou ze základních didaktických kategorií. V širším slova smyslu je chápána jako způsob dosažení nebo postup k výchovně vzdělávacímu cíli. [4]

Lze ji také chápat jako koordinovaný systém činností učitele a učebních aktivit žáka. [5]

Ve vyučování je pak vhodné používat a vhodně koordinovat různé výukové metody, protože jednostranné nadužívání jedné metody vede k nezájmu studentů nebo k jejich neúspěchu. [4]

V dalších podkapitolách se pokusíme charakterizovat rozdíl mezi tradičními a moderními metodami výuky a uvést jejich příklady. Ovšem zkoumaná literatura se příliš definicí těchto pojmů nezabývá, a pokud ano, definuje je velmi volně. Chápání těchto pojmů je navíc

časově podmíněno – co je moderní dnes, za pár let může být chápáno jako tradiční, ne-li zastaralé a překonané.

1.2.1 Tradiční metody výuky

Metody výuky jsou determinovány mimo jiné cíli výuky. A cíle výuky odpovídají historické době, ve které výuka probíhá. Např. středověké školství je proslulé memorováním jako typickou metodou výuky. Žák byl většinou nucen učit se poznatky nazpaměť, porozumění poznatkům bylo vedlejší. To odpovídalo nejhlavnějšímu cíli výuky té doby, „nacpat“ si do hlavy co nejvíce informací. Cíl to byl logický, knihy byly v té době velmi vzácné a dohledat neznámou či zapomenutou informaci bylo obrovským problémem.

Metody se postupem času proměňovaly. Různí reformátoři přinášeli do školství nové poznatky. U nás je znám především Komenský, který přinesl nový pohled na výuku a stanovil didaktické zásady.

Jedno však zůstalo dlouhou dobu nezměněno. Celá výuka je pevně řízena učitelem. Např. Průcha [6] vidí tradiční výuku zejména ve frontálním vyučování, komunikaci v tradiční výuce pak tvoří zejména prezentace učiva, opakování a zkoušení. Pecina [7] charakterizuje dosavadní pojetí výuky jako transmisivní, tedy že učitel je předavatelem hotových pravd. Čtrnáctová [8] uvádí, že tradiční výuka je založena na behaviorismu, orientovaná na cíl, učitel je v ní aktivním vedoucím, žáci jsou v ní pasivní, se sníženou odpovědností, vedení učitelem, musí dodržovat vnější kázeň. Z hlediska výkonu žáka se podle ní podobá individuálnímu sportu.

Tedy jednoduše shrnuto, tradiční výuka je založena zejména na frontální výuce s převládajícím výkladem, opakováním a zkoušením, s velkou rolí učitele a poměrně malou aktivitou žáků, determinovanou pokyny učitele. Dalšími častými znaky jsou poměrně hodně memorování, dril, diktování zápisů do sešitu.

Tyto tradiční metody často přetrvávají do současné doby a mají dosud řadu zastánců a praktikujících učitelů. Sám jsem při svém středoškolském studiu (maturoval jsem v roce 2008) zažil učitele, kteří nadiktovali do sešitu a pak z toho jen zkoušeli. Nazýval jsem je „zkoušející diktátoři“.

Jednou z příčin tohoto stavu je to, že učitelé napodobují své vlastní učitele. Petty [9] uvádí příklad učitelky práva, která se s velkým nadšením zúčastnila kurzu aktivního vyučování. Přesto při vlastní výuce střídavě přednášela nebo diktovala zápis s odůvodněním, že při

výuce práva to jinak nejde, že v tomto oboru je příliš mnoho učiva. Přičemž napodobovala její vlastní bývalé učitele, které obdivovala.

Příklad ilustruje fakt značné setrvačnosti ve školství. Zatímco v informatice se 20 let starý počítač považuje za zastaralý, mnozí kritikové tvrdí, že dnešní výuka se mnohdy příliš neliší od výuky před 100 lety, např. Šteffl [10]. Osobně jsem byl přítomen na celokrajském setkání Zlínského kraje k tématu „Kvalita ve školství“ v roce 2011, kde vystupoval i zmíněný Šteffl a dotázal se účastníků, převážně pracujících v oblasti školství, zda by si vybrali současnou nemocnici nebo nemocnici před 100 lety a poté zda by si vybrali současnou školu nebo školu před 100 lety. Zatímco nemocnici před 100 lety preferovalo minimum účastníků, u školy taktéž převažovala současná škola, ale již zdaleka ne tak výrazně.

I v pedagogické teorii najdeme zastánce tradičních metod. Maňák [11] uvádí, že tradiční výukové metody vždy najdou své místo ve školách, jsou ustálené, ověřené praxí a jsou tradiční základnou a předpokladem pro další rozvoj našeho školství. Nebo Pecina a Zormanová [12]: *„Ač je tradiční výuka často kritizována, je třeba si uvědomit, že transmisivní výuka má i v dnešní škole svůj význam, neboť pomocí tradiční výuky má žák látku utříděnou v uceleném systému.“*

Tito zastánci tradičních metod však mají i své tvrdé kritiky – Čapek [13] přirovnává v kritice děl těchto autorů tradiční výukové metody k valše a autory samotné ke *„kozlům píšícím příručku o pěstování salátu“*.

1.2.2 Příčiny snah o modernizaci

I když vývoj lidské společnosti se oproti středověku zrychloval celý novověk, velmi prudký rozvoj lze vidět po druhé světové válce. Příčin je pravděpodobně vícero. Neopominutelný je jistě především technologický rozvoj, zejména rozvoj informačních a komunikačních technologií.

Najít neznámou informaci je dnes s použitím internetového vyhledávače velmi často otázkou sekund. V případě více specializovaných informací může být čas delší, ale pokud informaci zná jakýkoliv odborník na planetě, není tak obtížné jej pomocí moderních technologií kontaktovat. To logicky mění cíle výuky – cílem přestává být namemorovat se co nejvíce informací. Roste role toho, aby se žák učil informace vyhledávat, pracovat s nimi, třídit je a v neposlední řadě aby se informacemi nezahltil. Tuto novou orientaci kurikula charakterizuje i tzv. Bílá kniha [14]. Nové kurikulum podle ní nemá být založeno na osvo-

jování co největšího počtu faktů. Nová orientace má spočívat např. na vyhnutí se zahlcení novými informacemi, umět kriticky myslet a hodnotit, umět se orientovat v různých situacích apod.

Další motivací k hledání nových metod výuky je neefektivita těch dosavadních. Mokrejšová [8] shrnuje závěry některých mezinárodních výzkumů, kde žáci z ČR sice vykazují vyšší znalosti, ale mají nižší úroveň praktických dovedností, malou orientaci v systému a jsou demotivovaní. Čapek [13] argumentuje, že proč by používal tradiční metody, když nebaví jeho žáky ani jeho samotného jako učitele. Také se objevuje zájem přitáhnout žáky k oborům, které jsou považovány za společensky potřebné, ale zájem o studium těchto oborů je mezi potencionálními žáky a studenty malý.

Zájem o zlepšení, případně zatraktivnění, výuky (často zejména matematiky, technických a přírodovědných předmětů) často projevují i státy či nadstátní organizace. Evropský unie podporuje řadu projektů takto zaměřených, např. ESTABLISH (Evropská věda a technologie v akci: Budování spojů s průmyslem, školou a domovem), S-TEAM (Vylepšené výukové metody pro učitele vědy), Fibonacci, PRIMAS (Podpora bádání ve výuce matematiky a vědy napříč Evropou) [8] nebo zde ve Zlínském kraji ve spolupráci se zahraničními partnery SEE („Závazek“ STEM pro Evropu, STEM – Věda, technologie, inženýrství a matematika) [15]. Na státní úrovni jsou podporovány např. TLRP (Výzkumný program učení a vyučování) v Británii nebo PTPO (Podpora technických a přírodovědných oborů) v ČR [8].

V neposlední řadě jsou příčinou modernizace nové didaktické poznatky. Pedagogická věda totiž též nespala a vyvíjela se. Petty [9] konstatuje, že v dnešní době máme mnohem více poznatků o vyučování a nejlepších výsledků je dosahováno aktivním učením pomocí náročných a zajímavých samostatných prací.

1.2.3 Moderní metody výuky

Jak se tedy změnila výuka v reakci na nové požadavky? Dömischová [16] uvádí, že moderní výuka by měla být motivující a efektivní, navazovat na realitu dnešního života, vytvářet předpoklady pro individuální rozvoj každého jednotlivého žáka, jeho samostatného, kreativního i kritického myšlení. Pecina [7] chápe jako nové konstruktivistické pojetí výuky, tedy že učitel vede žáky ve výuce k přemýšlení a získávání nových poznatků. Mezi moderní metody řadí především aktivizující metody (brainstorming, diskuzní metody, didaktické hry atd.) a komplexní metody (projekty, skupinová výuka atd.). Čapek [13] uvádí,

že jako moderní učitel nezkouší žáky u tabule ani orientačně, komunikace žáků má převahu nad verbálním projevem učitele a nepožaduje po žácích každou hodinu zapisovat do sešitu, který *„přestává být učebnicí číslo dvě, ale pracovním nástrojem“*.

Pokud bychom to měli shrnout, tak moderní metody jsou takové metody, které se snaží využít nových poznatků k zefektivnění, případně zatraktivnění výuky. Klíčovým znakem je aktivita žáka, který přestává být pasivním posluchačem. Často se více zapojují moderní technologie, více se ve výuce odráží praktický život.

V rámci této práce nelze prezentovat všechny moderní metody. Čapek ve své knize Moderní didaktika [13] popisuje více než 300 výukových metod. Další neustále vznikají. V roce 2016 na konferenci v Uherském Brodě prezentoval Prinssen [17] obrácené učení („reversed learning“), jehož snahou je propojit principy komerčního obchodu a nekomerčního vzdělávání a klade důraz na neformální učení („80 % naučeného se naučíme neformálně“). Různé obory se učí přímo v reálné školní továrně na balenou vodu. Dalším zajímavým příspěvkem z této konference byl příspěvek Ó Grádaigha [18], který ve výuce podporuje vytváření obsahu („content creation“) – tedy že si žáci sami pomocí moderních technologií vytváří vlastní materiály.

Jiným příkladem moderních metod může být badatelsky orientovaná výuka neboli IBSE. Mokrejšová [8] uvádí, že jde o výukový postup založený na vlastním zkoumání, že se se zaměřuje na diagnózu problému, experimentování, plánování a dalších činností a že badatelsky orientovaná a tradiční výuka nejsou protiklady a měly by být vzájemně kombinovány.

2 POŽADAVKY NA VÝUKU PROGRAMOVÁNÍ

Tato kapitola obsahuje zásadní informace z dokumentů, které determinují výuku programování na střední škole. Kapitola je zaměřena na výuku programování konkrétního oboru konkrétní školy, a to oboru Informační technologie na škole Obchodní akademie Kroměříž.

Podkapitola 2.1 se zabývá Rámcovým vzdělávacím programem oboru Informační technologie a podkapitola 2.2 Školním vzdělávacím programem tohoto oboru na Obchodní akademii Kroměříž. Podkapitola 2.3 shrnuje požadavky, které v informatických oborech stanoví Národní soustava kvalifikací.

2.1 Rámcový vzdělávací program

Rámcový vzdělávací program je státem vydaný pedagogický (kurikulární) dokument, který vymezuje závazné požadavky na vzdělávání v daném stupni a oboru. [19]

Rámcový vzdělávací program pro obor vzdělání 18-20-M/01 Informační technologie [19] (RVP) obsahuje několik požadavků, které by měl absolvent daného oboru v oblasti programování zvládat. RVP udává jednu z odborných kompetencí absolventa oboru „*programovat a vyvíjet uživatelská, databázová a webová řešení, tzn. aby absolventi algoritmizovali úlohy a tvořili aplikace v některém vývojovém prostředí; realizovali databázová řešení; tvořili webové stránky*“. Jedno z možných uplatnění absolventa dle RVP je programování a vývoj uživatelských, databázových a webových řešení, tedy práce programátora.

Výuka programování je v RVP zařazena ve vzdělávací oblasti Programování a vývoj aplikací (PVA). Učivo této oblasti se člení na pět částí: Algoritmizace, Strukturované programování, Úvod do objektového programování, Základy jazyka SQL a Tvorba statických a dynamických webových stránek. Požadavky na výsledky vzdělávání v této oblasti nejsou příliš vysoké – např. algoritmizovat úlohu, vytvářet jednoduché strukturované programy, používat jednoduché objekty apod. Minimální stanovený počet hodin pro tuto oblast je 8 týdenních a 256 celkem.

Mimo výše zmíněné oblasti je Algoritmizace zařazena i jako součást učiva vzdělávací oblasti Vzdělávání v informačních a komunikačních technologiích. Požadavky na výsledky vzdělávání jsou duplicitní s oblastí Programování a vývoj aplikací. Při pozornějším čtení RVP snadno zjistíme, že vůbec takřka celá oblast Vzdělávání v informačních a komunikačních technologiích je duplicitní s odbornými vzdělávacími oblastmi. Je to způsobeno tím, že oblasti všeobecného vzdělávání jsou pro celý stupeň jednotné, jak uvádí sám RVP.

Podařilo se mi objevit školy, které problém řeší tak, že tuto oblast rozdělí (dezintegrují) do odpovídajících odborných předmětů [20], což je samozřejmě možné [21] a dle mého názoru i správné. Ovšem řada škol prostě ponechá vzdělávací oblast jako předmět, jehož učivo se duplikuje s dalšími předměty. [22][23][24]

2.2 Školní vzdělávací program

Obor vzdělávání Informační technologie je na Obchodní akademii v Kroměříži oborem poměrně novým. Je vyučován od 1. 9. 2014 v čtyřletém denním studiu.

Školní vzdělávací program Informační technologie [22] byl tedy vytvořen nedávno. Programování se věnuje především předmět pojmenovaný stejně jako vzdělávací oblast z RVP, Programování a vývoj aplikací (PVA), který je zařazen ve druhém až čtvrtém ročníku. Ve druhém ročníku jsou vyučovány tematické celky Tvorba webových stránek, Algoritmizace a Strukturované programování, ve třetím ročníku Skriptování na straně klienta, Skriptování na straně serveru a Objektově orientované programování, ve čtvrtém ročníku pak Základy jazyka SQL, Definice objektů, Dotazy a modifikace dat a Webové aplikace a jejich propojení s databázemi. Předmět je vyučován 3 týdenní hodiny ve druhém a třetím ročníku a dvě týdenní hodiny ve čtvrtém ročníku. Algoritmizace je rovněž součástí výuky prvního ročníku v předmětu Informační a komunikační technologie. Je součástí širokého tematického celku, který má dotaci 10 hodin. Oba zmíněné předměty jsou vyučovány děleně.

V době vzniku tohoto textu byl na daném oboru vyučován jediný volitelný předmět z oblasti informačních technologií, Programování a tvorba webu. Tematické celky předmětu byly pouze opakováním předmětu Programování a vývoj aplikací. Na škole však proběhla debata o volitelných předmětech a bylo rozhodnuto upravit a rozšířit nabídku volitelných předmětů. Byl jsem požádán, abych vytvořil návrh nového předmětu zaměřeného na programování, který bych později sám vyučoval. Tvorba předmětu Seminář z programování je předmětem praktické části. Další předměty, vytvořené učiteli této školy, jsou Moderní technologie (tvořené tematickými celky Robotika, Programování mobilních aplikací, Modelování v grafických aplikacích a 3D tisk) a Seminář z tvorby webu a počítačové grafiky (jehož obsah je výstižně charakterizován již jeho názvem).

Všechny volitelné předměty jsou vyučovány dvě týdenní hodiny ve třetím a čtvrtém ročníku.

2.3 Kvalifikační standardy

Obsah této kapitoly vychází z [25].

Od roku 2005 realizuje MŠMT s partnery projekty týkající se Národní soustavy kvalifikací (NSK). V rámci NSK jsou zástupci zaměstnavatelů (např. skupinami odborníků v sektorových radách) vytvářeny kvalifikační a hodnotící standardy pro jednotlivé profese. Standardy poté prochází složitým schvalováním, do kterého jsou zapojeny reprezentace zaměstnavatelů či některá ministerstva.

Kvalifikační standard definuje, co má uchazeč umět, tedy požadované znalosti a dovednosti. Hodnotící standard říká, jak se tyto znalosti budou zkoušet. Doklad o úspěšném vykonání zkoušky má statut veřejné listiny. Prokazuje, že jeho držitel je kompetentní vykonávat dané povolání a jako takový je zaměstnavateli uznáván. Doklad však nemůže nahradit např. výuční list, ale v některých případech může jeho získání výrazně usnadnit.

V informatických oborech obsahuje NSK tři kvalifikace – Správce operačních systémů pro malé a střední organizace, Programátor (oba odpovídají úrovni 4 Evropského rámce kvalifikací – tedy úrovni středoškolského vzdělání s maturitou, i když jej nenahrazují) a Návrhář software (úroveň 5).

U Správce se pak v oblasti programování předpokládá, že ovládá odbornou způsobilost Základy programování skriptů a dávek, tzn. dovede srovnat nástroje pro skriptování, umí naprogramovat jednoduchou úlohu s pomocí skriptovacího nástroje, využívá znalosti základních příkazů operačního systému v dávkách a skriptech a orientuje se v anglicky napsaném manuálu a využívá jej k psaní skriptů.

Programátor by měl ovládat čtyři odborné způsobilosti: Analýza a algoritmizace praktických úloh, Tvorba programu ve vybraném prostředí, Tvorba uživatelského rozhraní a Ověření funkčnosti programu a testování optimálnosti algoritmu.

Návrhář software by pak měl být odborně způsobilý v následujících oblastech: Algoritmizace úlohy, Návrh databází, Tvorba schémat a diagramů s využitím jazyka UML, Principy programování, Optimalizace výkonnosti software, Zajištění bezpečnosti software, Použití SQL.

3 TVORBA VÝUKOVÝCH MATERIÁLŮ

Tato kapitola se zabývá výukovými materiály a jejich tvorbou. Přestože obdobným tématem se zabývá i část předchozí práce autora [1] a některé prvky mohou být podobné, kapitola rozhodně není pouhým opisem nebo citací z dané části původní práce.

V části 3.1 je definován výukový materiál a popsány různé kategorie výukových materiálů. Část 3.2 se zabývá jednotlivými fázemi tvorby materiálů. Část 3.3 popisuje učební úlohy a nastiňuje jejich kategorizaci. Podmínky, které na tvorbu výukových materiálů klade autorské právo, jsou uvedeny v části 3.4.

3.1 Co je výukový materiál

Lepil [26] charakterizuje výukové materiály jako všechna sdělení učební informace, která jsou verbální, grafická, obrazová a případně audiovizuální. Tato sdělení mohou mít tištěnou nebo elektronickou podobu (dodejme, že výjimečně se mohou objevit i ručně psaná sdělení, která Lepil opomíná). V širším pojetí sem lze zařadit i pomůcky pro demonstrační či žákovské experimenty.

Kocka [27] se zabývá knižními učebními pomůckami, mezi které řadí téměř veškeré tištěné výukové materiály, tzn. následující:

- **učebnice**, což je pomůcka obsahující metodicky zpracované učivo předepsané osnovami pro příslušný předmět, druh školy a zpravidla jeden ročník, učebnice se dělí na pracovní, programované a kombinované;
- **učební texty**, které obsahují učivo dle osnov, ale nejsou didakticky propracované jako učebnice a přechodně ji nahrazují;
- **pokusné učební texty**, které jsou využívány především na ověřování hypotéz pedagogického experimentu;
- **čítanky**, tedy systémově uspořádané soubory ukázek z vybraných děl;
- **didakticko-metodické texty**, mezi které patří pracovní listy, pracovní sešity a didaktické texty;
- a **učební příručky**, které se dělí na pravidla pravopisu, tabulky (matematické, fyzikální atd.), sbírky úloh, atlasy, slovníky, encyklopedie, kompendia, extrakty, sborníky, anály, letopisy, kroniky, průvodci, excerpty, repetitoria a bibliografie.

Aktuální materiály MŠMT [28] rozlišují zejména

- **učebnice**, což jsou didakticky zpracované texty a grafické materiály, které umožňují dosažení očekávaných výstupů vzdělávacích oborů vymezených RVP a využití tematických okruhů průřezových témat k rozvoji osobnosti žáka vymezených rámcovými vzdělávacími programy a směřují k utváření a rozvíjení klíčových kompetencí žáků, přičemž nejsou určeny ke znehodnocení jedním žákem pro další použití (například psaním, kreslením nebo rozstříháním);
- **učební texty**, mezi které jsou řazeny materiály, které jsou pro výuku nepostradatelné a doplňují učebnice, ale nemohou být součástí učebnic (odborné tabulky, pravidla českého pravopisu, normativní mluvnice, pracovní sešity tvořící jeden funkční celek s učebnicí, atlasy, pomůcky nahrazující běžné učebnice užívané při vzdělávání dětí a žáků se speciálními vzdělávacími potřebami);
- a **ostatní texty a materiály**, které volně doplňují učebnice nebo mají charakter cvičebnic nebo je možno je vytvářet ve variantách (pravopisná cvičení, diktáty a jazykové rozbory, sbírky úloh, pracovní sešity netvořící s učebnicí jeden funkční celek, slovníky, metodické příručky pro učitele, manuály k výuce software, ostatní pomocné textové, obrazové nebo číslicové materiály, audiovizuální materiály, multimediální programy pro PC).

Učebnice a učební texty mohou obdržet schvalovací doložku MŠMT. Školy však mohou na základě rozhodnutí ředitele používat i učebnice a učební texty bez doložky MŠMT, pokud obsahem vyhovují předpisům a pedagogickým a didaktickým zásadám. [29] Tedy doložky MŠMT jsou spíše doporučením pro školy. Ostatním materiálům a textům se doložka neuděluje [28].

3.2 Fáze tvorby výukových materiálů

Mladý [30] člení tvorbu výukových materiálů (učebnic) následovně:

- **vypracování osnovy učebnice** – učebnice by se měla navazovat na osnovy předmětu a nemůže se od nich odklánět, její rozsah se ale od osnov může lišit;
- příprava materiálů před tvorbou učebnice
    - **vypracování zásad tvorby** – řeší konstrukci tématu, která se dělí obvykle na název, úvod a základní definici, jádro, význam a uplatnění, kontrolní otázky a cvičení; dále na technické věci, jako číslování obrázků;

    - **příprava podkladů pro tvorbu učebnice** – jde především o shromažďování pramenů, ale i zjištění názorů učitelů, žáků, inspektorů i dalších a výsledků průzkumů a jiných experimentů;
- **harmonogram tvorby učebnice** – nutno vytvořit se zohledněním náročnosti jednotlivých témat, je významný především proto, že i zkušenému autorovi se může stát, že přestane registrovat náročnost jednotlivých témat, což samozřejmě podobu učebnice značně ovlivní;
- příprava a tvorba rukopisu;
- technická příprava, tisk apod.

3.3 Učební úlohy

Důležitou součástí moderní výuky jsou bezpochyby učební úlohy. Pro učební úlohu existuje velký počet definic. Např. Pedagogický slovník [31] charakterizuje učební úlohu jako každou pedagogickou situaci, která se vytváří proto, aby u žáků zajistila určitého učebního cíle. Všimněme si, že tato definice je poměrně podobná definici výukové metody. Jiné vymezení pochází od Tollingerové [32], která ji popisuje jako jazykový útvar, nebo promluvu, která se výslovně (verbálně), nebo svým kontextem (neverbálně), stává nositelem signálu „teď musím něco udělat“, na rozdíl od prosté zprávy, která je nositelem signálu „teď se něco dozvím“. Řadu definic dalších autorů obsahuje práce Netušilové [33].

Tollingerová [32] pak vytváří taxonomii učebních úloh, kde je řadí do pěti kategorií:

- úlohy vyžadující pamětní reprodukci poznatků,
- úlohy vyžadující jednoduché myšlenkové operace a poznatky,
- úlohy vyžadující složité operace a poznatky,
- úlohy vyžadující sdělení poznatků,
- a úlohy vyžadující tvořivé myšlení.

Tyto kategorie jsou pak dále členěny na 27 typů.

3.4 Autorské právo při tvorbě výukových materiálů

Výukové materiály mohou být chápány jako autorské dílo. Tvůrce těchto materiálů by si proto měl být vědom, jaké požadavky a omezení na něj klade autorské právo. Lepil [26] uvádí, že učitelé často neúmyslně autorská práva porušují. Bolenová – Sušická a Heinzová [34] pak uvádějí, že autorské právo je nejčastěji porušováno tak, že jsou rozmnožovány a

šířeny výňatky autorských děl (textů učebnic), přičemž škola či její zaměstnanci nesmí provádět toto rozmnožování a šíření pro potřebu žáků, nemají-li svolení autora. Údajně dokonce existují zprávy o případech, kdy žáci vydírali své učitele, kteří měli porušovat autorská práva [35].

V České republice je primárním zdrojem autorského práva Zákon č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) [36]. Zákon byl poměrně často novelizován, poslední účinná novelizace proběhla Zákonem č. 298/2016 Sb. Od 1. 7. 2017 nabude účinnosti Zákon č. 250/2016 Sb., o odpovědnosti za přestupky a řízení o nich [37], který také novelizuje autorský zákon. Dále v textu se budeme zabývat aktuálně účinnou verzí, tedy před novelizací Zákonem č. 250/2015 Sb.

Právem dílo užít se zabývá § 12. Dle odstavce 1 pak lze dílo užít bez oprávnění autora jen v případech stanovených autorským zákonem. To však má své výjimky a omezení, specifikované v díle 4. Pro tvorbu výukových materiálů je vhodné zejména se seznámit s § 31.

Z praxe je známo, že se vyskytly situace, kdy se výklad autorského práva jednotlivými institucemi lišil. V roce 2009 tvrdili zástupci MŠMT, že protizákonné může být i okopírování jediného cvičení na i/y. Agentura Dilia, zastupující mimo jiné autory učebnic, však vyjádřila odlišný názor. [38] Mezi učiteli údajně také koluje omyl, že autorským zákonem chráněný materiál lze použít, pokud je to pro výukové účely. [35] Aby se učitel vyhnul problémům či nejasnostem s autorským zákonem, radí někteří odborníci, aby si učitelé tvořili vlastní materiály. [39]

II. PRAKTICKÁ ČÁST

4 TVORBA OBSAHU NOVÉHO VOLITELNÉHO PŘEDMĚTU ZAMĚŘENÉHO NA PROGRAMOVÁNÍ

Jak již bylo uvedeno, byl jsem požádán o vytvoření nového volitelného předmětu pro Obchodní akademii Kroměříž, který by byl zaměřen na programování a který bych sám vyučoval. Vzhledem k tomu, že výukové materiály pro tento předmět jsou součástí práce, zařazuji do práce i proces tvorby tohoto předmětu.

Podkapitola 4.1 popisuje výběr názvu předmětu. V podkapitole 4.2 je shrnut výběr vzdělávacího obsahu předmětu. Podkapitola 4.3 se zabývá studijními materiály a materiálním zajištěním předmětu. V části 4.4 jsou obsaženy informace o schválení předmětu.

4.1 Název předmětu

Při výběru názvu předmětu hrálo roli několik faktorů. Určitou dobu jsem preferoval Pokročilé programování. Tento název vycházel z faktu, že předmět rozšiřuje výuku programování nad rámce předmětu Programování a vývoj aplikací. Nakonec se mi to nejevilo jako vhodné. Programování je poměrně náročnou disciplínou, což by mohlo mnohé žáky odradit samo o sobě. Slovo „pokročilé“ v nich mohlo evokovat, že se v předmětu mohou dostat k ještě složitějším problémům. Nakonec jsem vyhodnotil, že jednoduchost názvu předmětu prospěje a předmět jsem nazval Seminář z programování.

4.2 Vzdělávací obsah předmětu

Jedním z nejdůležitějších úkolů při tvorbě nového předmětu je stanovit si, jaké učivo bude v předmětu vyučováno. Pro výběr učiva jsem si stanovil několik kritérií:

- učivo by mělo být dostatečně srozumitelné a přiměřeně náročné pro žáka střední školy;
- učivo by se nemělo přímo krýt s učivem v jiných předmětech, mělo by ale na učivo jiných předmětů navazovat (zejména jde o předmět PVA);
- učivo by mělo být (pokud možno) zajímavé a atraktivní;
- učivo by mělo být pro žáka využitelné i v dalším životě (odborná praxe, další vzdělávání, zaměstnání);
- budoucí vyučující (kterým by měl být autor práce – viz výše) by měl mít dostatečné znalosti učiva.

Na základě těchto kritérií jsem si vytvořil seznam okruhů učiva, které by je naplňovaly. Témata jsem vybíral na základě svých vlastních znalostí o programátorském oboru. Vynechal jsem témata, která byla probírána v předmětu PVA. Vůbec jsem nezvažoval témata jako Teoretická informatika nebo Simulační programy, protože je považuji za pro žáka střední školy příliš náročné, vždyť i mnohý student vysoké školy se s nimi tvrdě potýká. Také jsem nezařadil témata, která by měla být procvičována během celého předmětu, jako Kvalita, čitelnost a optimálnost zdrojového kódu. K okruhům ve vytvořeném seznamu jsem poté hledal argumenty pro a proti. Na základě těchto argumentů okruhy seřadil dle vhodnosti zařazení do předmětu. Prvních pět jsem pak do předmětu zařadil. Vyhodnocení je v následující tabulce. Detailní specifikace vzdělávacího obsahu je součástí přílohy P1.

Tab. 1. Výběr učiva pro předmět Seminář z programování

Téma	**Pro**	**Proti**	**Zař.**
2D grafické programování (jednoduché hry)	Zábavné, atraktivní, využitelné v praxi, názorné		✓ ano
Významné algoritmy	Pochopení souvislostí, orientace v postupech řešení problémů		✓ ano
Abstraktní datové typy (ADT)	Často používané, pochopení souvislostí		✓ ano
Programování v ERP systému	Umět programovat v různých jazycích, znát praktické aplikace programování, soulad se zaměřením školy, uplatnění v Kroměříži	Praktické využití jen ve specifických případech	✓ ano
Skriptování v příkazové řádce	Vyžadováno na VŠ a u síťových/serverových administrátorů, mocný nástroj	Poměrně náročné	✓ ano
Strojově orientované programování (asem-	Pochopení vztahu mezi	Mnoho memorování instrukcí, asemblery	✗

blery)	HW a SW	dnes nahrazovány vyššími jazyky	ne
Práce se soubory specifického formátu	Využije téměř každý programátor	Lze se naučit samostatně, formáty jsou dobře zdokumentovány	✗ ne
Herní inteligence pro deskové hry	Zábavné	Malé využití	✗ ne
Kryptologie	Zábavné	Praktické využití jen ve specifických případech	✗ ne
Fungování překladačů	Pochopení souvislostí při překladu zdrojových kódů	Hraniční náročnost	✗ ne

4.3 Materiální zajištění předmětu a studijní materiály

V materiálu pro schválení nového předmětu bylo rovněž nutné popsat, co bude k výuce předmětu potřeba a z čeho se budou žáci učit. V materiálu nejsou uvedeny počítače, protože je jimi škola vybavena, je uveden pouze software.

V software jsou logicky zařazeny především překladače a vývojová prostředí. Za hlavní jazyk byl navržen C/C++, částečně by mohla být využita Java. K programování grafiky se pak jevila jako vhodná knihovna wxWidgets. K výuce skriptování je navrženo využít i operační systém Ubuntu. Dosud uvedený software lze získat bezplatně. Není uveden systém MS Windows, který je již na počítačích nainstalován. Jediným potencionálně placeným software, který je nutný k výuce programování v ERP systému, je ABRA Gen. Podařilo se však vyjednat bezplatné poskytnutí tohoto software od společnosti ABRA software.

Co se týče studijních materiálů, byly vybrány především weby zabývající se danou problematikou. Cena knih je v současné době poměrně vysoká, navíc web rychleji reaguje na technologické novinky. Tištěná literatura týkající se programování v ERP ABRA podle mých informací neexistuje. Dále je nutno počítat i s faktorem pohodlnosti žáků, kteří snáze kliknou na internetový odkaz, než aby navštěvovali knihovny či knihkupectví.

Detaily lze nalézt v příloze P1.

4.4 Schválení předmětu

Dle [19] je za tvorbu a schválení ŠVP odpovědný ředitel školy. ŠVP však musí být rovněž projednán ve školské radě [29]. Ředitel školy změnu schválil na jaře roku 2017. Školská rada změnu pravděpodobně projedná v průběhu května nebo června 2017.

5 NÁVRH VÝUKOVÝCH POSTUPŮ

Tato kapitola se zabývá přípravou výukových postupů a výukových metod, které budou využity v předmětu Seminář z programování. Návrh je zaměřen na učivo třetího ročníku, tedy především na programování v ERP systému a 2D grafické programování.

Cílem textu této kapitoly je popsat hlavní výukové metody pro výuku programování. Tedy nelze říci, že níže uvedené metody jsou jediné dobré a správné a že nelze použít vůbec nic jiného. Předpokládá se, že doplňkově mohou být využity ve výuce i jiné činnosti, např. didaktické hry.

Kapitola je rozdělena na dvě části, první se zabývá výkladem s diskuzí a demonstracemi, druhá praktickými úlohami.

5.1 Výklad s diskuzí a demonstracemi

Jak již bylo zmíněno v kapitole 1, výklad má v řadách pedagogické veřejnosti své zásadní odpůrce. Já jsem obdobně jako oni přesvědčen o tom, že výklad nemá být dominantní metodou výuky a že většinu informací, které lze předat výkladem, lze získat i praktickou zkušeností. Nicméně v oblasti programování získání této zkušenosti trvá měsíce práce na plný úvazek. Proto je nutné se o některých aspektech programování pobavit, něco si ukázat.

Proto byla v rámci práce vytvořena sada prezentací, které mají výkladu napomoci. Prezentace jsou pojaty velmi stručně, obsahují jen základní informace a mají být zároveň lehkou oporou a přehledem pro samostudium žáků. Jejich úkolem je i napomoct žákům, pokud dočasně ztratí pozornost, znovu „zachytit nit“ výkladu. Důvodem stručnosti prezentací je i to, že součástí výkladu budou především demonstrace jednotlivých postupů. Zde je zásadní role učitele přímo v hodině, aby dodržoval správné zásady pro didaktickou názornost. K demonstracím (i zobrazení prezentací) bude využit projektor, kterým jsou počítačové učebny školy vybaveny. Za důležité je považováno i propojit výklad s diskuzí. Proložení výkladu otázkami umožní kontrolovat pozornost žáků a zároveň je donutí trochu se zamyslet.

Zkušenost dále ukazuje, že v hodinách konaných v počítačové učebně je někdy velmi obtížné udržet pozornost. Řada žáků vnímá počítač jako zajímavou „hračku“, která je neustále nutí na něco klikat, surfovat na internetu či hrát hry, což snižuje pozornost k probíranému učivu.

Inspirací, která mě vedla k návrhu částečného řešení tohoto problému, bylo sledování výkladu některých pedagogů v neinformatických předmětech o tom, jak používají aktivizační výukové metody. Nabízela se otázka, zda jde něco podobného realizovat i v takto počítačově orientovaném předmětu. Lze v tomto předmětu vůbec donutit žáky vstát od počítačů? Napadla mě odpověď, že některé algoritmy lze přece předvést pohybem. Jako nejvhodnější se tato metoda jeví k předvedení řadících algoritmů (bubble sort, select sort, insert sort atd.). Příklady využití této metody jsou popsány v příloze P2. Zjednodušeně řečeno jde o to, že žáci tvoří řazené prvky a jiný žák nebo učitel organizuje jejich seřazení dle daného algoritmu.

5.2 Praktické úlohy

Protože programování je především praktická činnost, mají v jeho výuce praktické úlohy zásadní roli. Stejně jako není možné naučit se řídit auto bez vzetí volantu do rukou nebo naučit se vařit bez zapnutí sporáku, nelze se naučit programovat bez toho, aby si žák sám něco nenaprogramoval.

Úlohy jsou rozděleny na úlohy typu „naprogramuj“, typu „prověř“/„oprav“, úlohy na refaktorizaci kódu, vývojové diagramy. Poslední část je věnována diskuzi řešení úloh.

5.2.1 Úlohy typu „naprogramuj“

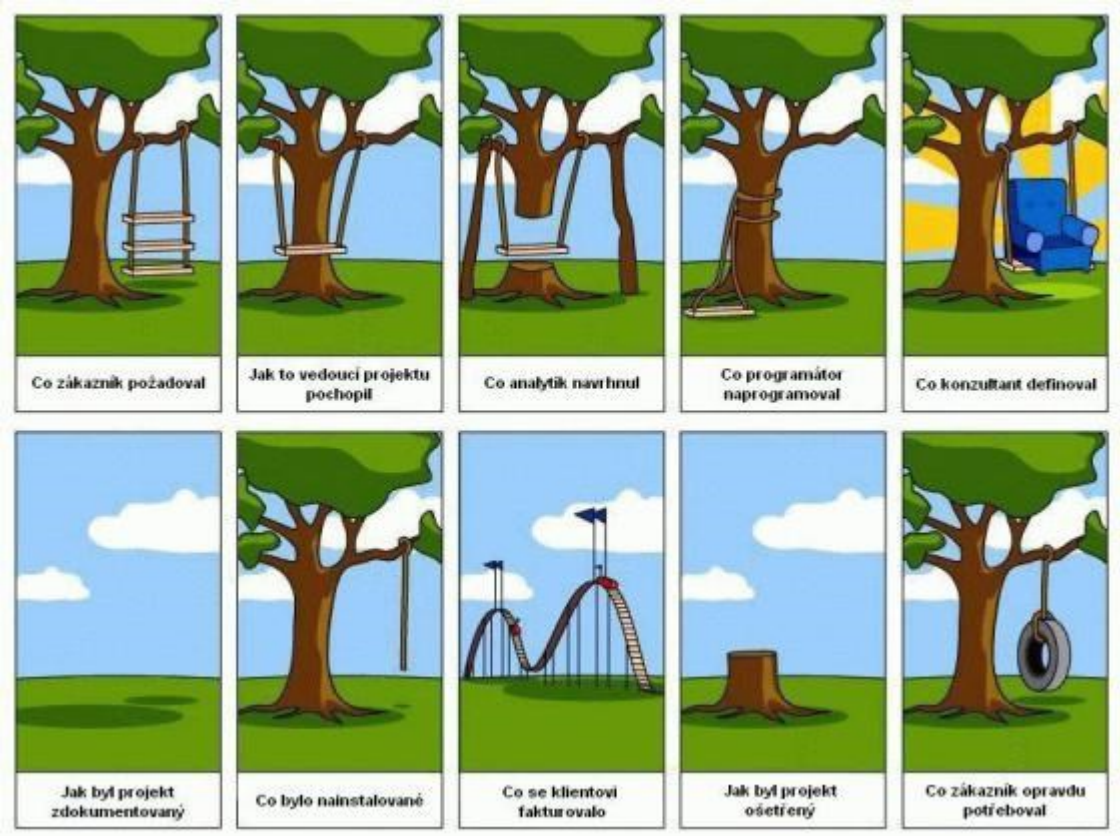
Tyto úlohy budou tvořit největší část úloh. Jde o úlohy, kdy se žákovi popíše, jak se má výsledný program chovat, co má dělat a žák toto chování převede do programovacího jazyka. Podobné činnosti patří k nejčastější náplni práce programátora. Několik zadání typu „naprogramuj“ je uvedeno v následující tabulce.

Tab. 2. Příklady zadání úloh typu „naprogramuj“ (vybráno z [1])

Zadání
Napište program, který z koeficientů kvadratické rovnice počítá její kořeny. Nezapomeňte řešit zvláštní případy (např. a == 0).
Vytvořte program, který generuje aritmetické příklady pro žáky první třídy (čísla v příkladech i výsledky příkladů by se měly pohybovat v intervalu <0; 20>, pouze operace sčítání a odčítání).
Napište funkci, která nalezne zadané slovo v osmisměrce. Osmisměrku reprezentujte jako

matici písmen. Pokuste se vytvořit program, který po nalezení všech zadaných slov v osmisměrce nalezne tajenku osmisměrky.

Tyto úlohy mohou být obměňovány. Dle vlastní zkušenosti je poměrně časté, že zadání programátora v soukromé firmě nejsou kompletní. Tzn. zadání sice něco vypovídá o požadavku zadavatele, ale k realizaci je potřeba od zadavatele zjistit doplňující informace. Situaci ilustruje známý obrázek.



Obr. 1. Průběh tvorby softwaru [40]

Proto budou zařazeny i úlohy, kdy učitel bude simulovat zadavatele, a žáci budou muset otázkami zjistit, co přesně mají dělat. Část úloh je koncipována tak, že žák sám je spolutvůrcem zadání (jednoduchá počítačová hra).

Komunikace mezi zadavatelem (učitelem) a zhotovitelem (žákem) může vypadat např. takto:

Zadavatel: „Vytvořte knihovnu s funkcemi na výpočet objemu a povrchu geometrických těles.“

Zhotovitel: „O jaká tělesa má jít?“

Zadavatel: „Minimálně krychle, kvádr, koule, válec, jehlan, kužel.“

Zhotovitel: „Z jakých údajů mají být tyto hodnoty vypočítány?“

Zadavatel: „Z délek stran.“

Zhotovitel: „Koule nemá stranu.“

Zadavatel: „U koule použijte poloměr, u válce a kuželu počítejte s výškou. Uvažujte pravidelné verze těchto těles.“

5.2.2 Úlohy typu „prověř“/„oprav“

I s úlohami tohoto typu se může programátor setkat ve svém zaměstnání. Každý programátor musí počítat s tím, že ve své práci bude opravovat a ladit nejen chyby vlastní, ale i chyby kolegů, např. na dovolené nebo již ve firmě nepracujících. Dle Hlavy [41] říká první Murphyho zákon programování, že neexistuje program bez chyb. A dodatek k tomuto zákonu, že odstranění chyby zanese do programu jinou, zákeřnější. O závažnosti problému softwarových chyb hovoří i fakt, že dle výzkumu Cambridge University jsou celosvětové roční náklady na debugování software 312 miliard dolarů a vývojáři software tráví 50 % svého programovacího času hledáním a odstraňováním chyb. [42]

Ke všemu mohu dodat vlastní zkušenost, že zákazníci často tvrdí, že v programu je chyba, ale nejsou schopni nebo ochotni přesně popsat, kde má programátor chybu hledat. Příklad: zákazník prohlásí, že chyba je u Lady. O kolik by měl programátor snazší práci, kdyby zákazník sdělil celé jméno dané osoby – Meruňková Vladimíra. Mnohdy se také zjistí, že chyba ve skutečnosti neexistuje – je způsobena chybnou prací zákazníka s programem (situaci opět naznačuje obr. 1 – zákazník požaduje něco jiného, než skutečně potřebuje).

Tyto údaje svědčí o tom, že zařazení úloh, kde budou žáci prověřovat funkčnost programu, zda program funguje správně či opravovat chyby v něm, je více než žádoucí. Tyto úlohy lze též užívat ve více variantách, např. je možné prověřovat zcela konkrétně popsanou chybu nebo prověřovat, zda vůbec nějaká chyba v programu je. Případně také odhadovat, k čemu má neznámý program sloužit.

Softwarové chyby jsou dvojího druhu (pokud nepočítáme nedodržení zadání, ale to může být způsobeno jak programátorem, tak nejasností zadání). Syntaktické chyby jsou chyby, kdy napsaný zdrojový kód je v rozporu se specifikací jazyka. Tyto chyby odhalí překladač, který zároveň nedovolí překlad a v případě kvalitního překladače zpravidla i navrhne vhodné řešení. Není proto nutné hledání takovýchto chyb příliš procvičovat. Zákeřnější jsou ovšem chyby sémantické. U těchto chyb nedochází k rozporu se specifikací jazyka,

ovšem program se nechová, jak má – může např. neočekávaně spadnout nebo dělat něco jiného, než má. Překladač tyto chyby nevyhodnotí jako bránící překladu a překlad provede (pokud nejsou v kódu žádné syntaktické chyby). Kvalitnější překladače ovšem u některých konstrukcí, které jsou pravděpodobně sémanticky chybné, vypisují varování. Na ta ovšem nelze na 100 % spoléhat, konstrukce jinak správná může být v určitém kontextu sémantickou chybou.

V následující tabulce jsou uvedeny příklady často se objevujících sémantických chyb.

Tab. 3. Časté sémantické chyby v jazyce C

Chyba	**Příčina chyby**	**Pravděpodobný následek**

`if (ch = 't')`
`{`
`  doSomething();`
`}`
	Přiřazení (=) místo porovnání (==)	Jiné chování než očekávané

`for (i = 0; i <= n; i++)` `  a[i] = countSth();`	V důsledku špatné ukončovací podmínky zápis mimo paměťový prostor programu	Pád programu

`x = 0;`
`//...`
`Z = y / x;`
	Dělení nulou	Pád programu

5.2.3 Úlohy na refaktorizaci kódu

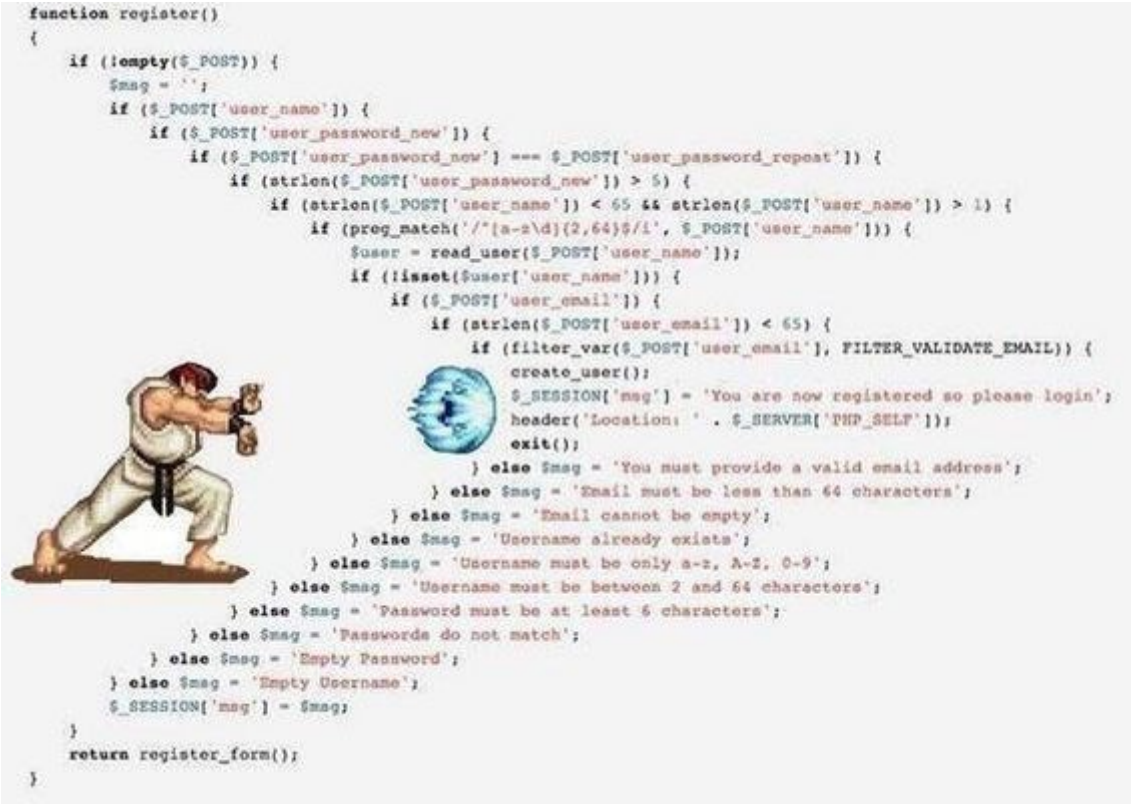
Ne vždy musí být špatný kód nutně chybný. Existují případy, kdy je program vlivem špatného naprogramování např. zbytečně pomalý. Někdy dokonce program běží správně, pracuje dostatečně rychle, ale problém je v tom, že kód je naprosto nečitelný. Přičemž v tomto kontextu je slovem nečitelný myšleno to, že kód je napsán nepřehledně a obtížně se v něm orientuje. Tyto problémy vychází z nedodržování konvencí pro správný styl zápisu zdrojového kódu, případně ze špatné aplikace těchto konvencí. O těchto konvencích již bylo popsáno mnoho, zde si vyjmenujeme příklady takových konvencí dle Šalouna [43]:

- jeden příkaz na jeden řádek;
- odsazování řádků;
- vynechávání řádku mezi logickými celky;
- rozlišování velkých a malých písmen;

- komentování kódu (co, jak, proč, kdy, kým);
- délka řádku s ohledem na šířku obrazovky;
- zalamování dlouhých řádků na logickém místě;
- rozumná délka jednotlivých logických bloků (metod, podprogramů, apod.);
- mezery ve vztahu k okolí;
- snažit se pro vše používat jen jeden jazyk (čeština – nepoužívat diakritiku, angličtina);
- nepoužívat příliš dlouhé identifikátory;
- dodržovat jednotná pravidla pro tvorbu názvů a identifikátorů;
- používat výstižné a odlišitelné názvy identifikátorů a metod (bez kolize);
- podmínky se snažit zapisovat v pozitivní formě.

Pokud vznikne požadavek na úpravu takového programu s nečitelným zdrojovým kódem, je velmi často lepší kód nejprve refaktorovat a teprve poté provést požadovanou úpravu.

Zparodovaný příklad špatného stylu zdrojového kódu ukazuje obrázek 2. Jde o častý příklad bezmyšlenkovité snahy o správné odsazování, která ovšem ústí v to, že se ze zdrojového kódu dosti špatně čte, co je podstatou algoritmu. Navíc pravá část kódu v těchto případech mnohdy přeteče mimo obrazovku. Využití konstrukcí if-else-if, spojek and/or nebo příkazů break/continue/return by algoritmus učinilo mnohem čitelnějším (povšimněme si, že často existuje více možností, jak učinit špatně čitelný kód přehlednějším). Podobný styl bývá nazýván dle zmíněného obrázku „styl bojovník“.



```
function register()
{
    if (!empty($_POST)) {
        $msg = '';
        if ($_POST['user_name']) {
            if ($_POST['user_password_new']) {
                if ($_POST['user_password_new'] === $_POST['user_password_repeat']) {
                    if (strlen($_POST['user_password_new']) > 5) {
                        if (strlen($_POST['user_name']) < 65 && strlen($_POST['user_name']) > 1) {
                            if (preg_match('/^[a-z\d]{2,64}$/i', $_POST['user_name'])) {
                                $user = read_user($_POST['user_name']);
                                if (!isset($user['user_name'])) {
                                    if ($_POST['user_email']) {
                                        if (strlen($_POST['user_email']) < 65) {
                                            if (filter_var($_POST['user_email'], FILTER_VALIDATE_EMAIL)) {
                                                create_user();
                                                $_SESSION['msg'] = 'You are now registered so please login';
                                                header('Location: ' . $_SERVER['PHP_SELF']);
                                                exit();
                                            } else $msg = 'You must provide a valid email address';
                                        } else $msg = 'Email must be less than 64 characters';
                                    } else $msg = 'Email cannot be empty';
                                } else $msg = 'Username already exists';
                            } else $msg = 'Username must be only a-z, A-Z, 0-9';
                        } else $msg = 'Username must be between 2 and 64 characters';
                    } else $msg = 'Password must be at least 6 characters';
                } else $msg = 'Passwords do not match';
            } else $msg = 'Empty Password';
        } else $msg = 'Empty Username';
        $_SESSION['msg'] = $msg;
    }
    return register_form();
}
```

Obr. 2. Ukázka zparodování špatného stylu zápisu programu [44]

Dále v tabulce je uvedeno několik příkladů špatného a správného kódu.

Tab. 4. Příklady kvalitního a nekvalitního kódu v jazyce Pascal

Kvalitní	**Nekvalitní**

`for i := 0 to n do`
`begin`
`  if not condition1 then`
`    continue;`
 	
`  if not condition2 then`
`    continue;`
 	
`  if not condition3 then`
`    continue;`
 	
`  doSomething();`
`end;`
	`for i := 0 to n do`
	`begin`
	`  if (condition1) then`
	`  begin`
	`    if (condition2) then`
	`    begin`
	`      if (condition3) then`
	`      begin`
	`        doSomething();`
	`      end;`
	`    end;`
	`  end;`
	`end;`

`readln(polomer, vyska);`
`objem := 2 * pi`
`  * polomer * polomer * vyska;`
	`readln(a, b);`
	`x := 2*PI*a*a*b;`
	`y := 2*PI*a*(2*a + b);`

```
povrch := 2 * pi * polomer
  * (2 * polomer + vyska);
writeln(objem, ' ' , povrch);
```

```
writeln(x, ' ' , y);
```

```
function isDivisible(
  dividend, divisor: integer
): boolean;
begin
  isDivisible := dividend
    mod divisor = 0;
end;

function isPrime(
  nr, lPrev: integer;
  prev: array of integer
): boolean;
var
  i: integer;
begin
  isPrime := true;
  for i := 0 to lPrev - 1 do
  begin
    if
      not isDivisible(nr, prev[i])
    then
      continue;

    isPrime := false;
    exit;
  end;
end;
```

```
function isprime(nr, lprev:
integer; prev: array of
integer): boolean; var i:
integer; begin isprime := true;
for i := 0 to lprev - 1 do
begin if not (nr mod prev[i] =
0) then continue; isprime
:= false; exit; end; end;
```

Zařazení úloh na opravu nekvalitního kódu tak sleduje dva výukové cíle. Prvním z nich je, aby žáci pochopili důležitost kvalitního zápisu programů. Druhým cílem je, aby se žáci, pokud na takový program narazí, byli schopni situaci řešit.

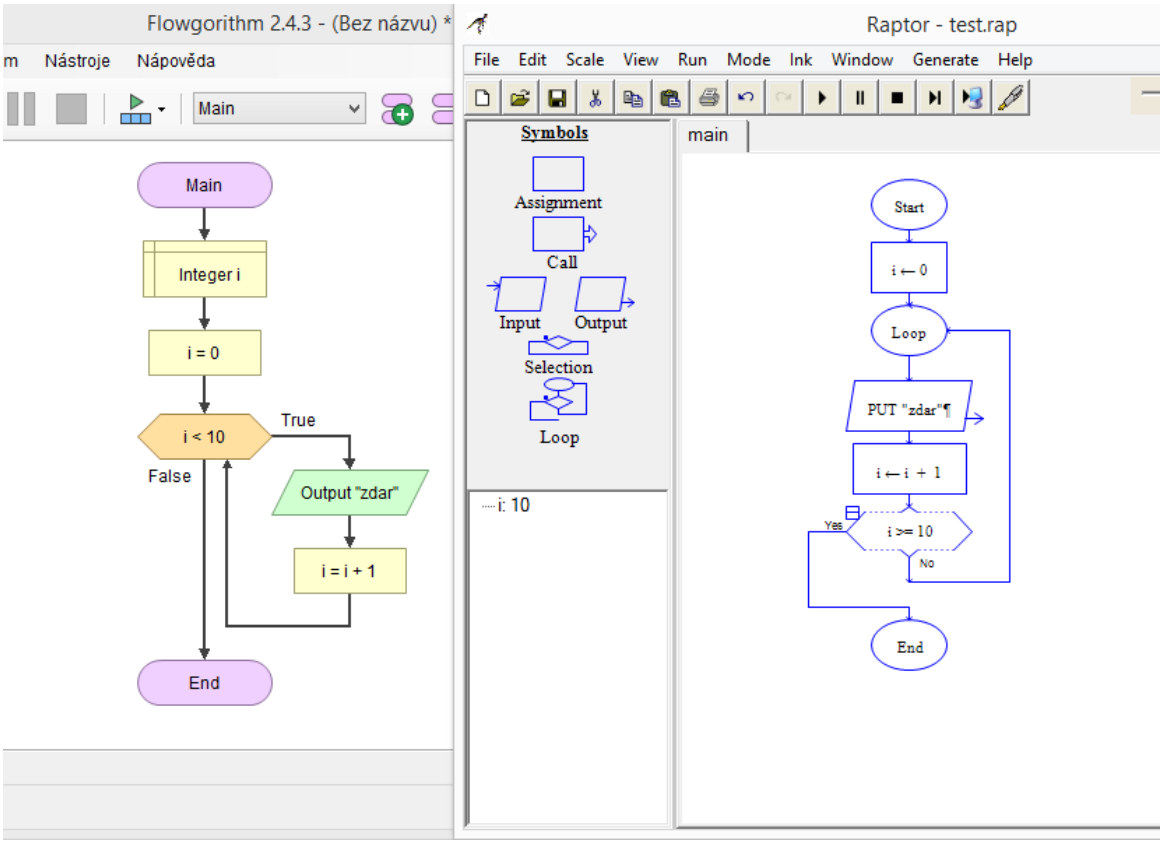
5.2.4 Vývojové diagramy

Vývojový diagram je grafická metoda vyjádření algoritmu, které je tvořeno bloky a jejich spojnicemi [45]. Chytil [46] o vývojovém diagramu říká, že vývojový diagram není nutností, ale je vhodný při skupinové práci, rozsáhlejším projektu nebo převodu programu

mezi dvěma programovacími jazyky. Osobně jsem zažil vyučující, kteří kladli na tvorbu vývojových diagramů velký důraz.

Mé poznatky ohledně výuky, kdy jsou žákům často zadávány úkoly, jejichž cílem je pouze vytvořit vývojový diagram na základě slovního popisu chování programu, jsou spíše negativní. Pro splnění úkolu většinou stačí jen nakreslit vývojový diagram na kus papíru. Tato činnost se ovšem z hlediska přípravy na reálné programování jeví jako nedostatečná. Reálné programování také není jen o napsání programu, ale i o jeho přeložení, spuštění a otestování. Tyto činnosti jsou s diagramem na kusu papíru naprosto vyloučeny a chyby v takovémto diagramu, které by v klasickém zdrojovém kódu odhalilo přeložení nebo první spuštění, často zůstávají neodhaleny, částečně i proto, že žák si řekne „nakreslil jsem, mám splněno“. Navíc ladění programu umožňuje především začínajícímu žákovi pochopit principy programování „zevnitř“. Někdy se také neodhalená chyba z jednoho diagramu šíří do dalších.

Vzhledem k výše uvedeným poznatkům považuji za nutné úkoly na tvorbu vývojových diagramů zařazovat obezřetně a raději méně než více. U pokročilejších žáků se nedá očekávat, že by jim tvorba diagramů dala více než tvorba zdrojového kódu. Diagramy snad mohou mít význam u žáků začínajících programovat (v případě Obchodní akademie Kroměříž žáci druhého ročníku) pro svou vysokou míru názornosti. Dnes již existují nástroje, které interpretují program zapsaný přímo vývojovým diagramem. Mezi tyto nástroje patří Flowgorithm [47], Raptor [48] nebo Visual Logic [49], viz obrázek. Při použití těchto nástrojů lze program do jisté míry testovat a ladit a z tohoto důvodu je jejich využití vhodnější než kreslení diagramů na papír.



Obr. 3 Ukázka vývojového diagramu v programech Flowgorihm a Raptor

Pro vyváženost nutno dodat, že stejně ochuzené jako kreslení vývojových diagramů na papír je i psaní zdrojových kódů na papír. Taktéž jsou žáci ochuzeni o překlad, testování i ladění.

5.2.5 Diskuze řešení praktických úloh

Netriviální úlohy z oblasti programování většinou nemají jen jedno řešení. Řešení může být poměrně mnoho. Některá řešení mohou být (téměř) rovnocenná, jindy je každé řešení vhodné pro jinou situaci a někdy se dá říct, že jedno řešení je výrazně vhodnější pro všechny situace než jiné nebo naopak jedno řešení je výrazně méně vhodné.

Vezměme si jako příklad řazení dat. Řadicích algoritmů existuje poměrně hodně, např. na webu algoritmy.net [50] je uvedeno 15 algoritmů. Algoritmy quick sort a heap sort lze z hlediska teoretické průměrné časové složitosti považovat za rovnocenné. Jsou však mezi nimi určité rozdíly. Quick sort je dle empirických studií sice v průměru rychlejší, ale za to má horší maximální časovou složitost – tedy pokud chceme co nejrychlejší řazení, jeví se jako vhodnější quick sort, pokud je řazení kritické a nesmí v žádném případě překročit

určitou mez, je logická volba heap sort [51]. Oproti tomu algoritmy bogosort a bozosort nemají žádnou výhodu a vždy lze nalézt vhodnější řešení.

Zadáme-li více studentům stejnou úlohu, u níž je velmi pravděpodobné, že různí studenti vypracují řešení, která budou mít výrazné rozdíly v principech, je vhodné, aby si žáci navzájem svá řešení odprezentovali a v případě odlišností byly důkladně prodiskutovány výhody a nevýhody jednotlivých řešení. Zkušenost s různými způsoby řešení může být důležitou inspirací pro řešení dalších úloh.

6 VYTVOŘENÍ KURZU V SYSTÉMU MOODLE

Tato kapitola se zabývá systémem Moodle a jeho využitím ve výuce programování. V kapitole 6.1 je systém charakterizován a v kapitole 6.2 jsou popsány možnosti přizpůsobení Moodlu pro výuku programování.

6.1 Charakteristika systému Moodle

Na Obchodní akademii Kroměříž je od 1. 1. 2015 využíván systém Moodle. [52] I proto je součástí práce vytvoření modulu právě do tohoto systému. Dalším důvodem je to, že většina materiálů tvořených v rámci této práce je vytvářena v elektronické podobě. Důležité je také to, že většina praktických činností v tématice předmětu probíhá na počítači a proto je vhodné využít k ucelené práci e-learningový systém.

Systém Moodle je známé a hojně užívané vzdělávací e-learningové prostředí. Jde o svobodný a otevřený systém. Je chráněn autorským právem, ale je možné jej volně šířit a modifikovat. Pro uživatelskou práci stačí domácí či přenosný počítač a webový prohlížeč (do Moodlu lze ovšem nahrát i soubory, které vyžadují další software). Autorem Moodlu je Martin Dougiamas [53], který je dodnes ředitelem společnosti Moodle Pty. Ltd. [54] Moodle vychází z principů sociálního konstrukcionismu. [53]

Sociální konstrukcionismus je směr v psychologii a v psychoterapii. Základní teze tohoto systému jsou, že lidé vytvářejí realitu především prostřednictvím jazyka, že člověku je třeba rozumět v kontextu jeho společnosti a že ani psychologii nelze oddělovat od společnosti, ve které se vyvíjí. Proto výsledky výzkumů psychologie jsou ne pouze produktem výzkumníka, ale je výsledkem „vyjednávání“ mezi zkoumajícím a zkoumaným. [55]

Z hlediska Moodlu považuje Drlík [53] ohledně sociálního konstrukcionismu za významné, že se učící se aktivně zapojuje do výuky a vzdělávací obsah je chápán z jeho pohledu. Učitel se pak má stát moderátorem, který vzdělávané spíše usměrňuje, ne jen předavatelem znalostí.

Přestože z výše uvedeného odstavce by se mohlo téměř zdát, že Moodle je použitelný výhradně na některé moderní metody výuky, není tomu tak. Moodle lze využívat mnoha rozmanitými způsoby, přičemž jej lze využít i jako podporu pro zcela tradiční výuku.

6.2 Přizpůsobení Moodle pro výuku programování

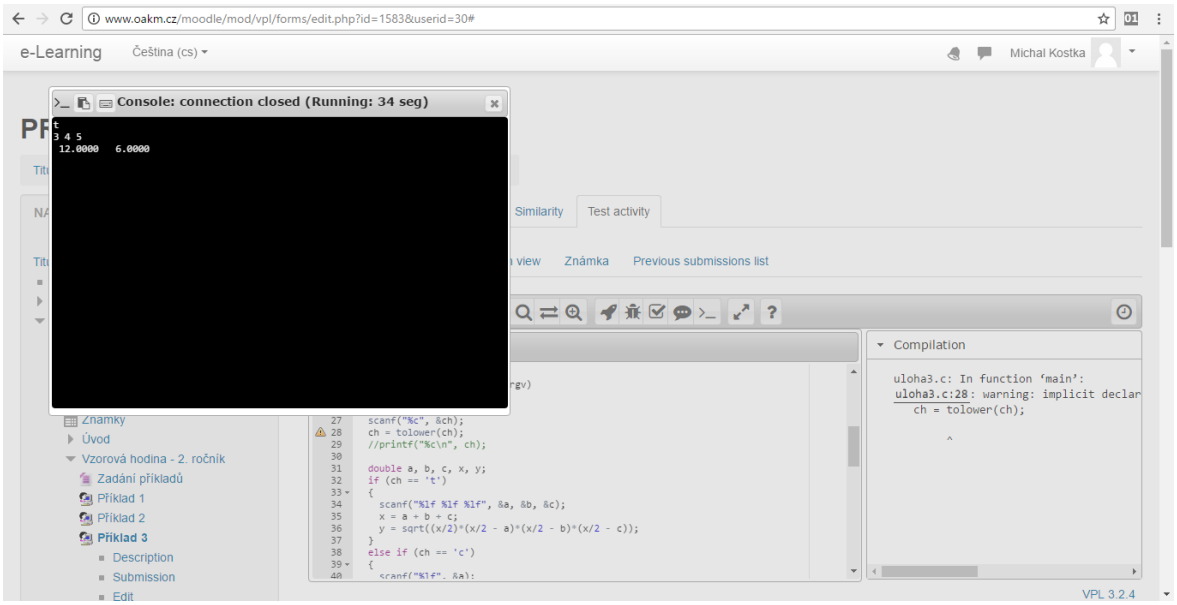
Výuka programování má na rozdíl od výuky dalších předmětů svá specifika. Jelikož Moodle jako takový byl navržen pro výuku obecně, jeví se jako vhodné zvažovat, nakolik lze systém uzpůsobit požadavkům pro tuto oblast.

Již se vyskytlo několik prací, jejichž úkolem bylo vytvořit modul pro výuku programování. Jde např. o práce Vaškové [56], Jury [57] a Benoviče [58].

Vzhledem k předpokladu snadné udržitelnosti je ovšem vhodné využít některý z registrovaných modulů.

Existuje např. modul CodeRunner. CodeRunner umožňuje klást dotazy, ve kterých student odpovídá kódem v nějakém programovacím jazyce. [59] CodeRunner se ovšem příliš nehodí na tvorbu větších a složitějších programů. Jeho síla je spíše v testování jednodušších funkcí.

Nejlepší předpoklady však ukázal modul VPL (Virtuální laboratoř programování). VPL umožňuje žákům editovat a sestavovat zdrojové kódy a programy přímo v prohlížeči. Učitel tak nemusí stahovat řadu souborů do počítače, může je testovat též přímo v prohlížeči. Při využití modulu tak odpadají časté spory typu „na mém počítači to šlo, jen u Vás to nejede“. Pokud učitel nechce testovat ručně, může využít automatizované testy. Modul obsahuje též nástroje pro kontrolu plagiátorství. Je podporována široká řada programovacích jazyků (Ada, C, C++, C#, Fortran, Haskell, Java, Matlab/Octave, Pascal, Perl, PHP, Prolog, Python, Ruby, Scheme, SQL and VHDL). Systém též automaticky rozpozná jazyk podle přípony souboru. Modul je možné užívat bezplatně. [60]



Obr. 4. Modul VPL v Moodlu Obchodní akademie Kroměříž

Modul VPL byl do Moodlu Obchodní akademie Kroměříž na mou žádost nainstalován správcem sítě dne 19. 4. 2017.

7 OVĚŘENÍ POSTUPŮ

Kapitola se zabývá ověřením postupů navržených k výuce programování. První část se zabývá přípravou ověření, poté je popsán průběh ověření, ve třetí části jsou shrnuty výsledky dotazníkového šetření a v poslední části je ověřování vyhodnoceno.

7.1 Příprava ověření

Doba na tvorbu této práce nedává příliš velký prostor na ověření navržených postupů. Autor práce v současné době nevyučuje a nemá tedy možnost věnovat ověřování mnoho hodin výuky. Navíc navržený volitelný předmět má být vyučován teprve v budoucnu. Bylo proto nutné zvážit, co a jakým způsobem bude ověřeno.

K ověření bylo vybráno následující:

- obsahové zaměření nového volitelného předmětu;
- využití modulu VPL v Moodlu k výuce programování;
- některé popsané výukové metody.

K ověření výukových metod se podařilo dojednat výuku dvakrát dvou (výuka odborných předmětů je zpravidla rozdělena na dvě skupiny) vzorových hodin ve druhém ročníku oboru Informační technologie na Obchodní akademii Kroměříž. V takto vymezeném čase nelze ověřovat všechny výukové metody. Proto byly k ověření vybrány zejména:

- pohybové znázornění algoritmů;
- úlohy typu „prověř“/„oprav“;
- úlohy na refaktorizaci kódu.

Výběr vycházel z názoru, že jiné metody se k ověření hodí méně. Výklad je metoda, jejíž hodnocení může být velmi subjektivní z důvodu, že kvalita výkladu značně záleží na kvalitě osoby provádějící výklad. Navíc se ve výuce užívá poměrně často, stejně jako úlohy typu „naprogramuj“. Proto byly zvoleny méně frekventované metody.

Při pohybovém znázornění měly být demonstrovány vybrané řadicí algoritmy v rámci výuky algoritmizace. Dále byly připraveny čtyři praktické úlohy na ověření. Zdrojové kódy, které mají žáci prověřit, opravit nebo refaktorovat, byly napsány v jazyce C, ve kterém se žáci učí programovat. Náročnost zadání je poměrně nízká, což vychází z faktu, že žáci se dosud ve výuce programování daleko nedostali (dle informací od vyučujícího Tomáše Valacha se dostali teprve k podmínkám).

První úloha obsahovala neznámý kód. Žáci měli zjistit, k čemu kód slouží a případně navrhnout vylepšení programu. Správné řešení bylo, že program určuje přestupnost zadaného roku. Vylepšení spočívalo ve spojení podmínek, čímž by došlo ke zjednodušení kódu.

Druhá úloha obsahovala porovnání dvou proměnných typu double. Do jedné byla vložena hodnota 0,1 a do druhé hodnota 0,01 + 0,09. Žáci měli rozhodnout, zda program funguje správně (po spuštění programu se čísla pochopitelně nerovnala), případně navrhnout opravu. Podstatou správné odpovědi je, že tato čísla, která jsou ve zdrojovém kódu zapsána v desítkové soustavě, nelze jednoznačně (bez zaokrouhlení nebo oseknutí) převést do dvojkové soustavy, ve které pracuje výsledný program a že porovnání čísel s plovoucí desetinnou čárkou tedy na dnešních architekturách počítačů dává nepřesné výsledky. Praktickým řešením může být napsat si vlastní porovnávací funkci. Tato funkce za shodná prohlásí ta čísla, pro která platí, že absolutní hodnota jejich rozdílu je menší než povolená chyba, tedy matematickým zápisem $|a - b| < E \rightarrow a = b$, kde a a b jsou porovnávaná čísla, E je vhodně zvolená povolená chyba (např. 10^{-9}).

Další úloha obsahuje neznámý kód s chybou a cílem je určit smysl kódu a chybu opravit. Chyba spočívá v (několikanásobné) záměně porovnání (==) za přiřazení (=). Smyslem kódu je pak počítat obvod a obsah zvoleného geometrického útvaru, případně objem a povrch zvoleného geometrického tělesa.

Poslední úloha je bonusová. Je spíše určena pro zabavení se rychlejších žáků, nepočítá se s tím, že ji zvládnou všichni žáci, znalosti potřebné k pochopení kódu dosud neprobírali. Úloha obsahuje kód generující příklady na převody číselných soustav. Nejprve je generováno zadání, poté řešení. Výstupy programu byly využity učiteli Obchodní akademie právě k výuce převodů číselných soustav. Úkolem žáků je pouze pochopit účel programu. Je jim i napovězeno, že se s výstupy programu setkali.

Připravené soubory i jejich řešení (řešení bylo v době ověřovací hodiny skryto) jsou součástí kurzu v Moodle. Pro každou úlohu byla navíc připravena činnost typu VPL.

K vyhodnocení byl připraven dotazník pro žáky a dotazník pro jejich vyučujícího, který vzorové hodiny sledoval. Dotazníky využívaly jak uzavřené, tak otevřené otázky. Dotazníky (i se shrnutím pozdějších odpovědí) jsou v přílohách P3 a P4.

7.2 Realizace ověření

Nejprve proběhlo ověření pohybového znázornění algoritmů v rámci výuky algoritmizace. Vždy polovina ze skupiny (třídy jsou pro výuku odborných předmětů rozděleny na dvě skupiny, viz výše) byla vyzvána, aby se postavila do řady. Jeden žák měl ostatní seřadit podle výšky tak, že bude udávat pokyny, kam se mají žáci postavit. V obou skupinách se řadící žák rozhodl vybírat nejvyššího žáka z neseřazených, kterého posílal na kraj neseřazené skupiny. Žákům tedy bylo vysvětleno, že aniž by jej znali, tak zcela intuitivně použili algoritmus. Dále, že algoritmus, který použili, se nazývá select sort (v češtině někdy řazení výběrem). Poté byla k postavení se do řady vyzvána druhá polovina skupiny. Autor práce je seřadil jiným algoritmem, aby demonstroval, že na řešení některých úloh, v tomto případě řazení, existují různé algoritmy. V první skupině byl využit algoritmus bubble sort (bublinové řazení). Několikanásobnou výměnu nejvyššího žáka se sousedem (charakteristickou pro tento algoritmus) jeden ze žáků komentoval slovy: „*Nebylo by jednodušší jej rovnou postavit na konec?*“ Ve druhé skupině byl demonstrován algoritmus insert sort (řazení vkládáním), který se obešel bez komentářů. Toto bylo využito jako odrazový můstek k dalšímu vysvětlování algoritmů.

Ověření praktických úloh se ukázalo jako komplikovanější záležitost. Oproti předpokladu autora se ukázalo, že žáci nemají v Moodlu založené vlastní účty. A žák, který do kurzu přistupuje jako host, nemůže úlohy modulu VPL vidět. Původní představa, že žáci budou pracovat přímo v modulu VPL, tak nemohla být realizována.

Žákům bylo na začátku hodiny vysvětleno, kde si mají stáhnout příklady a že pokud by v nich náhodou narazili na neznámou konstrukci, mohou se na ni zeptat.

První skupina tedy řešení realizovala ve Visual Studiu, prostředí, ve kterém se učí programovat. Přesto se projevila nedostatečná zkušenost žáků s tímto prostředím, kdy žáci měli problém samostatné soubory naimportovat do Visual Studia tak, aby je mohli přeložit. Vysvětlení, jak mají žáci soubory ve Visual Studiu překládat, zabralo značnou část hodiny. Proto se žáci víceméně zabývali jen prvními dvěma příklady. Přitom nikdo z žáků nebyl schopen ani částečně zformulovat smysluplné řešení.

Ve druhé skupině byl proto využit online překladač na [61]. Pravděpodobně i díky tomu byla práce s druhou skupinou úspěšnější, žáci se zabývali třemi úlohami. S mírnou dopomocí se někteří žáci byli schopni dopracovat aspoň k částečnému řešení. U první úlohy po nápovědě, že jde o něco s roky, byli schopni říct, že každý čtvrtý rok je přestupný. U druhé

úlohy jeden z žáků prohlásil, že jde o „něco s dvojkovou soustavou“. U třetí úlohy stejný žák prohlásil, že „jsou počítány rozměry trojúhelníka“.

Žákům v obou skupinách bylo na konci hodiny vysvětleno, jak a dokdy mají vyplňovat a předat anonymní dotazník. Další dotazník byl předán jejich vyučujícímu Tomáši Valachovi, který byl hodinám přítomen.

7.3 Výsledky dotazníkového šetření

Data z dotazníků byla sumarizována následujícím způsobem:

- U uzavřených (polouzavřených) otázek s předem vymezenými možnostmi bylo sečteno, kolik žáků vybralo jednotlivé možnosti a vypočten procentuální podíl těchto žáků, a to jak v rámci jednotlivých skupin, tak za obě skupiny dohromady.
- U otázky 4, kdy měli dotazovaní ohodnotit („oznámkovat“) jednotlivé tematické okruhy, byl vypočten průměr hodnocení pro každý tematický okruh, a to jak v rámci jednotlivých skupin, tak za obě skupiny dohromady. Jeden žák uvedl jednou hodnocení 6, vzhledem k tomu, že to příliš neovlivnilo výsledné hodnocení (tematický okruh i tak dosáhl nejlepšího hodnocení), bylo i toto hodnocení započítáno.
- U otevřených otázek se zjišťovalo, které odpovědi mají shodnou podstatu, např. u otázky 2 všechny odpovědi, které akcentovaly důraz na výhody z hlediska zaměstnání nebo pracovního uplatnění, byly zahrnuty do kategorie odpovědí „práce“. Nekonkrétní odpovědi typu „nevím“ nebyly kategorizovány. Tyto otázky nebyly sumarizovány v rámci jednotlivých skupin, ale jen za obě skupiny dohromady.
- Odpovědi vyučujícího nebyly sumarizovány s žákovskými odpověďmi.

Dotazníky odevzdalo 27 žáků. Z toho 12 žáků odevzdalo dotazník až po urgenci. Dva žáci odevzdali zcela identický dotazník (obě dvě kopie dotazníku byly do hodnocení zařazeny). Odevzdané dotazníky jsou součástí CD přiloženého k diplomové práci. Statistika odevzdaných žákovských dotazníků je v příloze P3. V příloze P4 jsou pak odpovědi vyučujícího.

Z dotazníků plyne, že 5 žáků si zvolilo jako volitelný předmět Seminář z programování). Podle Tomáše Valacha šlo i tak o druhý nejatraktivnější předmět (žáci měli na výběr ze šesti předmětů). Nejčastějšími argumenty, které žáci uváděli ve prospěch předmětu, bylo využití v práci a naučení se programovat. V neprospěch předmětu byla uváděna především

složitost a náročnost programování. Zhruba polovina žáků si zvolila jiný předmět z oblasti IT.

Co se týče témat předmětu, žáci přes zjevné upozornění, že nejde o stanovení pořadí, v řadě případů udělili všech pět možných hodnocení, takže jakoby pořadí stanovili. Proto se průměrná známka všech tematických celků pohybuje v poměrně malém rozmezí, a to 2,3 až 3,11. Malý rozdíl v průměrné známce i pohled na jednotlivé dotazníky naznačuje, že žáci mají rozdílné pohledy na to, která témata jsou přínosná. Nejlépe je hodnoceno 2D grafické programování. S odstupem cca 0,4 následují Významné algoritmy a Skriptování. Nejméně populární jsou Abstraktní datové typy. To potvrzuje i následná volná otázka, kdy odpověď preferující tvorbu her se opakovala 4x.

Jelikož modul VPL si žáci nemohli nakonec vyzkoušet, otázky 6 a 7, které se týkaly tohoto modulu, se staly bezpředmětnými. Proto zde byla většina odpovědí „nevím“.

U demonstrace algoritmů pohybem se ukázalo, že většina žáků (téměř 60 %) považuje tento způsob výuky za přínosný. Dalších 22 % uvádělo adjektiva zábavný a názorný. Pouze tři žáci hodnotili tuto metodu jako nudnou a zbytečně zdržující. Hodnocení neužitečný uvedl jeden žák. Tedy převažuje kladné hodnocení této metody nad negativním.

Zkoumání neznámých kódů bylo hodnoceno spíše negativně. Pouze jeden žák označil činnost za rozhodně přínosnou. 37 % pak činnost hodnotilo jako spíše přínosnou. Přesně třetina hodnotila činnost jako spíše nepřínosnou a čtyři žáci jako rozhodně nepřínosnou. Zde byl zaznamenán velký rozdíl mezi skupinami, kde první skupina hodnotila výuku významně hůře.

Vyučující žáků na drtivou většinu otázek dotazníku odpovídal velmi pozitivně. Jediná negativní hodnocení spočívala v tom, že vedení navrženého volitelného předmětu nemusí být snadné, a v tom, že tématu abstraktní datové typy udělil hodnocení 3, u ostatních témat udělil lepší hodnocení.

7.4 Vyhodnocení ověřování

Ze samotného průběhu ověřovacích hodin byly vyvozeny následující poznatky:

- Ověřovací praktické úlohy byly pro dané žáky příliš složité, přesto, že programovací konstrukce v nich obsažené žáci znali (a pokud některé neznali, např. operátor modulo, mohli se zeptat). Je možné, že úlohy typu „prověř“/„oprav“ je z tohoto dů-

vodu vhodnější zařazovat teprve tehdy, až si žáci důkladněji osvojí schopnost pracovat s programovacím jazykem a programovacím prostředím.

- Z výše uvedeného plyne také fakt, že učitel by při zadávání úloh měl být důkladně seznámen se schopnostmi svých žáků a měl by důkladně zvažovat, jaké úlohy jsou pro žáky vhodné a zvládnutelné. S tím je spojen názor, že časté střídání učitelů programování není vhodné, protože krátkodobý učitel nemůže být důkladně seznámen se znalostmi svých žáků.
- Vyučující informatických předmětů by se měl důkladně připravit na technologie, se kterými bude ve výuce pracovat, a další podmínky výuky. Tento poznatek je poměrně známý a toto ověřování jej jednoznačně potvrdilo. Pokud vznikne problém s technikou, v ostatních ohledech sebelépe připravená hodina informatického předmětu nemusí naplnit svůj účel.
- Ve výuce programování je nutné kromě výuky programovacího jazyka a programátorského myšlení věnovat přiměřenou pozornost i práci s vývojovými prostředími, aby se v nich žáci dobře orientovali.

Vyhodnocení dotazníků pak výše uvedené předpoklady potvrdilo. Z dotazníků rovněž vyplynulo, že záporné hodnocení prověřování neznámých kódů vychází zejména z problémů při přípravě hodiny a následných zmatků v hodině a možná i zatím nedostatečných zkušeností žáků spíše než z toho, že by úlohy tohoto typu byly zcela nevhodné. Jak uvedl jeden z žáků: „*Kdybychom tomu více rozuměli, tak si myslím, že by to bylo lepší…*“

Ukázalo se, že i pohybové znázornění algoritmů považují žáci za přínosné, i když ne zcela zábavné. Na druhou stranu jsou i žáci, které to vyloženě nebaví.

Žáci oboru IT také považují programování za přínosné z hlediska práce, ale za složité. Témata, která by se chtěli učit v programování, si ovšem zřejmě nevybírají z pracovního hlediska ani z hlediska náročnosti (je otázkou, nakolik jsou schopni náročnost témat vyhodnotit v době, kdy teprve začínají programovat), ale spíše preferují (zdánlivou) zábavu – to ukazuje výraznější preference tématu tvorby her, což považuji za spíše náročnější téma. Nicméně potvrdil se předpoklad, že půjde o téma pro žáky atraktivní.

ZÁVĚR

Práce se zabývala výukou programování na střední škole. Byly charakterizovány rozdíly mezi tradičními a moderními metodami výuky. Ukazuje se, že existují jak příznivci, tak odpůrci tradičních metod. Dále byly popsány požadavky na výuku programování z hlediska ŠVP na Obchodní akademii Kroměříž a RVP a také, jaké má požadavky na kvalifikované pracovníky v informatických oborech Národní soustava kvalifikací. Také byly nastíněny principy tvorby výukových materiálů včetně pohledu legislativních dokumentů, jako jsou školský a autorský zákon či některé vyhlášky ministerstev.

Praktická část práce byla zaměřena na výuku programování v oboru IT na Obchodní akademii Kroměříž. Byl navržen nový volitelný předmět Seminář z programování. Zařazení předmětu do ŠVP bylo schváleno ředitelem školy a předmět se pravděpodobně začne vyučovat od příštího roku. Rozhodující bude zájem žáků, ten by měl podle aktuálních informací být dostatečný. Nutno říci, že předmět prozatím nepatří k mezi žáky nejatraktivnějším ani nejméně atraktivním. Jeho atraktivitu snižuje náročnost, kterou žáci předpokládají. Na druhou stranu žáci projevili názor, že by předmět mohl být užitečný z hlediska jejich pracovního uplatnění a některé z nich zajímá zejména téma tvorby her, které je součástí předmětu.

Byla vytvořena metodika výuky (nejen) pro tento předmět. Spolu s touto metodikou byly vytvořeny prezentace, několik zadání úkolů a vzorových řešení, zejména pro třetí ročník navrženého volitelného předmětu. Na zadáních a vzorových řešeních zejména pro tematický celek Programování v ERP systému bude v případě otevření předmětu nutné ještě pracovat, protože autor v době tvorby práce neměl k dispozici ERP systém ABRA Gen s licencí na skriptování. V rámci metodiky bylo mimo jiné navrženo využít k výuce některých algoritmů pohyb žáků.

Veškeré materiály byly umístěny do Moodlu Obchodní akademie. Do Moodlu Obchodní akademie byl na žádost autora práce nainstalován modul VPL, který značně usnadňuje odevzdávání a hodnocení žákovských prací.

Ověření ukázalo, že pohybové znázornění algoritmů pokládají žáci za přínosné. Nepotvrdila se obava, že to většina žáků bude považovat za pouhou otravu, i když to nepovažují ani za velkou zábavu. Ověřování dále ukázalo, že úlohy na prověření cizího kódu není vhodné zařazovat v době, kdy se žáci teprve učí programovat a že ve výuce programování je nutné být důkladně seznámen s předpoklady žáků pro řešení dané úlohy a vědět, čemu byli dříve

vyučováni, s čím mají zkušenosti. Také je nutné prověřit, zda technika, která bude při výuce, je k řešení vhodná a nakolik s ní žáci umí pracovat. Pokud žáci neporozumí smyslu úlohy nebo např. nastanou komplikace s překladačem, je velmi těžké udržet zájem žáků o úlohu a žáci pak úlohu hodnotí jako zbytečnou a nepřínosnou.

Dá se předpokládat, že výsledky práce budou využity a dále rozvíjeny pro potřeby výuky na Obchodní akademii Kroměříž.

SEZNAM POUŽITÉ LITERATURY

[1] KOSTKA, Michal. *Sbírka příkladů z programování pro střední školy*. Brno, 2012. Závěrečná práce doplňujícího pedagogického studia. Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikační technologií. Vedoucí práce Veronika Svobodová.

[2] KAŠPÁRKOVÁ, Svatava. *Učení a vyučování*. Zlín: Univerzita Tomáše Bati ve Zlíně, 2013, 1 online zdroj (123 s.). ISBN 978-80-7454-298-5. Dostupné také z: http://hdl.handle.net/10563/25822

[3] RAMBOUSEK, Vladimír. *Materiální didaktické prostředky* [online]. V Praze: Univerzita Karlova, Pedagogická fakulta, 2014 [cit. 2017-03-22]. ISBN 978-80-7290-664-2. Dostupné z: http://vzdelavani-dvpp.eu/download/opory/final/23_rambousek.pdf

[4] ČANDÍK, Marek a Štefan CHUDÝ. *Didaktika informatiky*. Zlín: Univerzita Tomáše Bati ve Zlíně, 2005, 133 s. Učební texty vysokých škol. ISBN 8073182858.

[5] MAŇÁK, Josef a Vlastimil ŠVEC. *Výukové metody*. Brno: Paido, 2003, 219 s. ISBN 80-7315-039-5.

[6] PRŮCHA, Jiří. *Moderní vzdělávací technologie*. Praha: Vysoká škola J. A. Komenského, 2003, 93 s. ISBN 80-86723-01-1.

[7] PECINA, Pavel. *Moderní výukové metody a postupy* [online]. [cit. 2017-03-16]. Dostupné z: http://www.janwe.cz/getFile/id:2706

[8] ČTRNÁCTOVÁ, Hana a Olga MOKREJŠOVÁ. Tvorba výukových materiálů pro střední školy. Praha: Conatex-Didactic Učební pomůcky, 2013. ISBN 978-80-87936-02-3.

[9] PETTY, Geoffrey. *Moderní vyučování*. 6., rozš. a přeprac. vyd. Praha: Portál, 2013, 562 s. ISBN 978-80-262-0367-4.

[10] RODRIGUEZ, Veronika a Ondřej ŠTEFFL. *Expert živě: Na školách přetrvává Rakousko-Uhersko* [online]. 2009 [cit. 2017-03-16]. Dostupné z: https://zpravy.aktualne.cz/domaci/expert-zive-na-skolach-pretrvava-rakousko-uhersko/r~i:article:652397/

[11] MAŇÁK, Josef. *Alternativní metody a postupy*. 1. vyd. Brno: Masarykova univerzita, 1997. 90 s.

[12] PECINA, Pavel a Lucie ZORMANOVÁ. *Metody a formy aktivní práce žáků v teorii a praxi*. 1. vyd. Brno: Masarykova univerzita, 2009, 147 s. ISBN 978-80-210-4834-8.

[13] ČAPEK, Robert. *Moderní didaktika: lexikon výukových a hodnoticích metod*. Praha: Grada, 2015. Pedagogika. ISBN 978-80-247-3450-7.

[14] Ministerstvo školství, mládeže a tělovýchovy. *Národní program rozvoje vzdělávání. Bílá kniha*. Praha: Ústav pro informace ve vzdělávání – nakladatelství Tauris, 2001. ISBN 80-211-0372-8. Dostupné také z: http://www.nuv.cz/uploads/nuv/strategicke/Bila_kniha_2001.pdf

[15] NĚMCOVÁ, Martina. JAK ZATRAKTIVNIT PŘÍRODOVĚDNÉ A TECHNICKÉ OBORY NA STŘEDNÍCH ŠKOLÁCH? *Zkola: Informační a vzdělávací portál Zlínského kraje* [online]. [cit. 2017-03-26]. Dostupné z: https://www.zkola.cz/management/omsrlz/mezinarodnispolupracezlk/Stranky/STEM-projekt-Zlinskeho-kraje.aspx

[16] DÖMISCHOVÁ, Ivona. *Projektová výuka: moderní strategie vzdělávání v České republice a německy mluvících zemích*. Olomouc: Univerzita Palackého v Olomouci, 2011, 212 s. Monografie. ISBN 978-80-244-2915-1.

[17] PRINSSEN, Theo. *Superb Teaching: Reversed learning* [online]. Koning Willem I College, 2016 [cit. 2017-03-16]. Dostupné z: http://www.konferenceub.cz/wp-content/uploads/2016/10/3P_Superb-Teaching_Theo-Prinssen.pdf

[18] Ó GRÁDAIGH, Seán. *Mobility in Teacher Education: A Case Study* [online]. National University of Ireland, Galway, 2016 [cit. 2017-03-16]. Dostupné z: http://www.konferenceub.cz/wp-content/uploads/2016/10/Se%C3%A1n-%C3%93-Gr%C3%A1daigh.pdf

[19] *Rámcový vzdělávací program pro obor vzdělání 18-20-M/01 Informační technologie* [online]. Národní ústav odborného vzdělávání, Praha, 2007 [cit. 2017-03-16]. Dostupné z: http://zpd.nuov.cz/RVP/ML/RVP\%201820M01\%20Informacni\%20technologie.pdf

[20] BERNÁT, Jiří. *Školní vzdělávací program studijního oboru 18-20-M/01 Informační technologie: zaměření oboru: IT specialista* [online]. Praha: Střední Prů-

myslová Škola na Proseku, 2013 [cit. 2017-03-16]. Dostupné z: http://www.sps-prosek.cz/?wpdmact=process&did=NDcuaG90bGluaw

[21] *Metodika tvorby školních vzdělávacích programů SOŠ a SOU* [online]. Praha: Národní ústav pro vzdělávání, školské poradenské zařízení a zařízení pro další vzdělávání pedagogických pracovníků, 2012 [cit. 2017-03-16]. ISBN 978-80-87652-05-3. Dostupné z: http://www.nuv.cz/file/320_1_1/

[22] *Školní vzdělávací program studijního oboru 18-20-M/01 Informační technologie*. Kroměříž: Obchodní akademie Kroměříž, 2013 [cit. 2017-03-16].

[23] *Školní vzdělávací program Informační technologie* [online]. Jihlava: Střední průmyslová škola Jihlava, 2009 [cit. 2017-03-16]. Dostupné z: http://www.sps-jia.cz/blue/PDF/ITsvp.pdf

[24] *Školní vzdělávací program 18-20-M/01 informační technologie - počítačové sítě* [online]. Kralupy nad Vltavou: Střední odborná škola a Střední odborné učiliště Kralupy nad Vltavou, 2010 [cit. 2017-03-16]. Dostupné z: http://www.sosasoukralupy.cz/res/data/009/001159.pdf

[25] *Národní soustava kvalifikací* [online]. NÚV a TREXIMA, 2014 [cit. 2017-05-07]. Dostupné z: https://www.narodnikvalifikace.cz/

[26] LEPIL, Oldřich. *Teorie a praxe tvorby výukových materiálů: zvyšování kvality vzdělávání učitelů přírodovědných předmětů* [online]. Olomouc: Univerzita Palackého v Olomouci, 2010 [cit. 2017-03-22]. ISBN 978-80-244-2489-7. Dostupné z: http://zvyp.upol.cz/publikace/lepil.pdf

[27] KOCKA, Michal. *Tvorba učebníc: Súbor predpisov*. Bratislava: Slovenské pedagogické nakladateľstvo, 1981.

[28] *Směrnice náměstka ministra pro vzdělávání ministerstva školství, mládeže a tělovýchovy k postupu a stanoveným podmínkám pro udělování a odnímání schvalovacích doložek učebnicím a učebním textům a k zařazování učebnic a učebních textů do seznamu učebnic*. Praha: Ministerstvo školství, mládeže a tělovýchovy, 2013, č.j. MSMT-34616/2013. Dostupné také z: http://www.msmt.cz/file/32170/download/

[29] *Zákon č. 561/2014 Sb., o předškolním, základním, středním, vyšším odborném a jiném zdělávání (školský zákon), ve znění účinném od 1. 1. 2017 do 31. 8. 2017.*

In: Sbírka zákonů. 22. 4. 1998. ISSN 1211-1244. Dostupný také z: http://www.msmt.cz/file/39574/download/

[30] MLADÝ, Karol. *Tvorba a výroba učebníc*. Bratislava: Slovenské pedagogické nakladateľstvo, 1988.

[31] PRŮCHA, Jan, Eliška WALTEROVÁ a Jiří MAREŠ. *Pedagogický slovník*. 3., rozš. a aktualiz. vyd. Praha: Portál, 2001. ISBN 80-717-8579-2.

[32] TOLLINGEROVÁ, Dana a Antonín MALACH. *Metody programování*. Hradec Králové: Pedagogická fakulta, 1973. Učební texty vysokých škol.

[33] NETUŠILOVÁ, Miloslava. *Učební úlohy z fyzické geografie ve výuce zeměpisu na ZŠ*. Brno, 2008. Diplomová práce. Masarykova univerzita. Pedagogická fakulta. Vedoucí práce Eduard Hoffman.

[34] BOLENOVÁ – SUŠICKÁ, Kateřina a Radka HEINZOVÁ. *AUTORSKÁ PRÁVA - 2. webinář programu Škola na dotek* [online]. [cit. 2017-04-13]. Dostupné z: http://www.skotek.cz/wp-content/uploads/2014/04/AUTORSK%C3%81-PR%C3%81VA_z%C3%A1kladn%C3%AD_podklady.pdf

[35] Omyly týkající se autorských práv. *Otevřené vzdělávání* [online]. [cit. 2017-04-13]. Dostupné z: http://otevrenevzdelavani.cz/prirucka/omyly-tykajici-se-autorskych-prav/

[36] Zákon č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon). In: *Sbírka zákonů*. Dostupné také z: https://portal.gov.cz/app/zakony/download?idBiblio=49278&nr=121~2F2000~20Sb.&ft=pdf

[37] Zákon č. 250/2016 Sb., o odpovědnosti za přestupky a řízení o nich. In: *Sbírka zákonů*. Dostupné také z: http://www.mvcr.cz/soubor/zakon-c-250-2016-sb.aspx

[38] Učitelé pozor - kopírování učebnic porušuje zákon. *Česká televize* [online]. [cit. 2017-04-13]. Dostupné z: http://www.ceskatelevize.cz/ct24/domaci/1384130-ucitele-pozor-kopirovani-ucebnic-porusuje-zakon

[39] Kopírování učebnic může být nelegální. *Učitelské noviny* [online]. 2003 [cit. 2017-04-13]. Dostupné z: http://www.ucitelskenoviny.cz/?archiv&clanek=4786&PHPSESSID=0f25f5ca140cecb2f5e2a53fa17d088d

[40] HÁZE, Petr. Dostaň to, co chceš aneb úskalí externího IT vývoje. *SVĚTBYZNYSU.cz* [online]. 2013 [cit. 2017-04-25]. Dostupné z: http://www.svetbyznysu.cz/2013/07/dostan-to-co-chces-aneb-uskali-externiho-it-vyvoje/

[41] HLAVA, Tomáš. Chyby v softwaru. *Testování softwaru* [online]. 2011 [cit. 2017-04-25]. Dostupné z: http://testovanisoftwaru.cz/tag/chyba-softwaru/

[42] Financial Content: Cambridge University study states software bugs cost economy $312 billion per year. *University of Cambridge* [online]. 2013 [cit. 2017-04-25]. Dostupné z: http://insight.jbs.cam.ac.uk/2013/financial-content-cambridge-university-study-states-software-bugs-cost-economy-312-billion-per-year

[43] ŠALOUN, Petr. *Základy programování: Zdrojový kód, dokumentace, týmová práce* [online]. [cit. 2017-04-25]. Dostupné z: http://homel.vsb.cz/~s1a10/educ/ZP/prednasky/11-zdroj-kod-dokumentace.pdf

[44] Code Indentation - Street Fighter Style. In: *Pinterest* [online]. [cit. 2017-04-25]. Dostupné z: https://s-media-cache-ak0.pinimg.com/564x/41/d6/46/41d646a4b4ca17151c0a0ef6876859d5.jpg

[45] *Vývojové diagramy* [online]. [cit. 2017-04-27]. Dostupné z: https://is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=65549

[46] CHYTIL, Jiří. Vývojové diagramy - 1. díl. *Programujte.com* [online]. 2005 [cit. 2017-04-27]. Dostupné z: http://programujte.com/clanek/2005080105-vyvojove-diagramy-1-dil/

[47] *Flowgorithm* [online]. [cit. 2017-04-27]. Dostupné z: http://flowgorithm.org/

[48] *Welcome to the RAPTOR home page* [online]. [cit. 2017-04-27]. Dostupné z: http://raptor.martincarlisle.com/

[49] *Presenting Visual Logic 2.2.10* [online]. [cit. 2017-04-27]. Dostupné z: http://www.visuallogic.org/

[50] NECKÁŘ, Jan. *Algoritmy.net* [online]. 2016 [cit. 2017-05-08]. Dostupné z: https://www.algoritmy.net/

[51] MORRIS, John. *Comparing Quick and Heap Sorts* [online]. 1998 [cit. 2017-05-08]. Dostupné z: https://www.cs.auckland.ac.nz/software/AlgAnim/qsort3.html

[52] *Moodle - Obchodní akademie Kroměříž* [online]. [cit. 2017-04-20]. Dostupné z: http://www.oakm.cz/moodle/

[53] DRLÍK, Martin. *Moodle: kompletní průvodce tvorbou a správou elektronických kurzů.* Brno: Computer Press, 2013. ISBN 978-80-251-3759-8.

[54] Moodle Pty. Ltd. *FindTheCompany* [online]. [cit. 2017-04-20]. Dostupné z: http://listings.ftb-companies-au.com/l/102052826/Moodle-Pty-Ltd-in-Perth-WA

[55] PLHÁKOVÁ, Alena. *Dějiny psychologie*. Praha: Grada, 2006. Psyché (Grada). ISBN 80-247-0871-x.

[56] VAŠKOVÁ, Veronika. *Automatické hodnocení úkolů v kurzech programování*. Zlín: Univerzita Tomáše Bati ve Zlíně, 2005, 26 s. Dostupné také z: http://hdl.handle.net/10563/36287. Univerzita Tomáše Bati ve Zlíně. Fakulta technologická, Institut řízení procesů a aplikované informatiky. Vedoucí práce Dulík, Tomáš.

[57] JURA, Pavel. *Moduly eLearning systému Moodle pro potřeby výuky na UTB ve Zlíně*. Zlín: Univerzita Tomáše Bati ve Zlíně, 2006, 56 s., 6 s. příloh. Dostupné také z: http://hdl.handle.net/10563/1060. Tomas Bata University in Zlín. Faculty of Applied Informatics, Ústav aplikované informatiky. Vedoucí práce Dulík, Tomáš.

[58] BENOVIČ, Miroslav. *Automatické hodnocení úkolů v kurzech programování*. Zlín: Univerzita Tomáše Bati ve Zlíně, 2011, 49 s. Dostupné také z: http://hdl.handle.net/10563/17413. Tomas Bata University in Zlín. Faculty of Applied Informatics, Ústav automatizace a řídicí techniky. Vedoucí práce Dulík, Tomáš.

[59] LOBB, Richard a Tim HUNT. Question types: CodeRunner. *Moodle* [online]. [cit. 2017-04-20]. Dostupné z: https://moodle.org/plugins/qtype_coderunner

[60] *VPL, the Virtual Programming lab for Moodle* [online]. [cit. 2017-04-20]. Dostupné z: http://vpl.dis.ulpgc.es/

[61] COMPILE AND EXECUTE C ONLINE. *Tutorialspoint: SIMPLE EASY LEARNING* [online]. [cit. 2017-05-06]. Dostupné z: http://tutorialspoint.com/compile_c_online.php

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

2D	Two-dimensional, dvojrozměrný.
3D	Three-dimensional, trojrozměrný.
ADT	Abstraktní datový typ.
Apod.	A podobně.
Atd.	A tak dále.
Cit.	Citováno.
Č.	Číslo.
ERP	Enterprise resource planning, plánování podnikových zdrojů, dnes většinou označuje software vykonávající tuto činnost.
ESTABLISH	European Science and Technology in Action: Building Links with Industry, Schools and Home, Evropská věda a technologie v akci: Budování spojů s průmyslem, školou a domovem (mezinárodní projekt).
Hod.	Hodina, hodinový.
HW	Hardware.
IBSE	Inquiry Based Science Education, výuka vědy založená na bádání, jinak badatelsky orientovaná výuka.
IT	Informační technologie.
Ltd.	Limited, zde společnost s ručením omezeným (dle australského práva).
Moodle	Modular Object-Oriented Dynamic Learning Environment, modulární objektově orientované dynamické prostředí pro učení.
MS	Microsoft.
MŠMT	Ministerstvo školství, mládeže a tělovýchovy.
Např.	Například.
NSK	Národní soustava kvalifikací.
PHP	Hypertext Preprocessor/Personal Home Page, hypertextový preprocesor/osobní domácí stránka (skriptovací jazyk).

PRIMAS	Promoting Inquiry in Mathematics and Science Education across Europe, Podpora bádání ve výuce matematiky a vědy napříč Evropou (mezinárodní projekt).
PTPO	Podpora technických a přírodovědných oborů (český projekt).
Pty.	Proprietary, druh obchodní společnosti dle australského práva
PVA	Programování a vývoj aplikací (vzdělávací oblast RVP a zároveň předmět oboru IT na Obchodní akademii Kroměříž).
RAPTOR	Rapid Algorithmic Prototyping Tool for Ordered Reasoning, Rychlý algoritmický prototypovací nástroj pro uspořádané uvažování (programovací prostředí založené na vývojových diagramech).
RVP	Rámcový vzdělávací program.
S.	Strana.
Sb.	Sbírka zákonů ČR.
SEE	STEM Engagement Europe, „Závazek“ STEM pro Evropu (vzdělávací projekt Zlínského kraje a partnerů).
SOŠ	Střední odborná škola.
SOU	Střední odborné učiliště.
SQL	Structured Query Language, strukturovaný dotazovací jazyk (nejčastěji používaný jazyk pro práci s databázemi).
S-TEAM	Science-Teacher Education Advanced Methods, Vylepšené výukové metody pro učitele vědy (mezinárodní projekt).
STEM	Science, Technology, Engineering and Mathematics, věda, technologie, inženýrství a matematika, zpravidla myšleny předměty či obory této oblasti.
SW	Software.
ŠVP	Školní vzdělávací program.
TLRP	Teaching and Learning Research Programme, Výzkumný program učení a vyučování (britský projekt).

Tzn.	To znamená.
Tzv.	Takzvaný.
UML	Unified Modeling Language, unifikovaný modelovací jazyk.
VHDL	VHSIC Hardware Description Language, jazyk pro popis hardware VHSIC.
VHSIC	Very High Speed Integrated Circuit, vysokorychlostní integrovaný obvod.
VPL	Virtual Programming Lab, Virtuální laboratoř programování (modul pro systém Moodle).
VŠ	Vysoká škola.

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: NÁVRH NOVÉHO VOLITELNÉHO PŘEDMĚTU SEMINÁŘ Z PROGRAMOVÁNÍ

Název vyučovacího předmětu:	Seminář z programování
Kód a název oboru vzdělání:	18-20-M/01 Informační technologie
Dosažený stupeň vzdělání:	střední vzdělání s maturitní zkouškou
Délka a forma vzdělání:	čtyřleté denní
Počet vyučovacích hodin za studium:	123
Platnost:	od 1. 9. 2017 počínaje třetím ročníkem

Pojetí vyučovacího předmětu

Obecné cíle předmětu

Cílem předmětu je rozšířit znalosti a dovednosti žáka v oblasti programování nad rámec předmětu PVA. Žáci se učí prakticky aplikovat poznatky z předmětu PVA, rozšiřují si teoretické znalosti v oblasti algoritmů, osvojují si schopnost programovat v různých jazycích a dozvídají se nové informace o programátorské profesi.

Charakteristika učiva

Učivo se skládá z několika tematických celků. Část tematických celků slouží k obeznámení se s praktickými aspekty programování, část k prohloubení teorie algoritmizace.

Pojetí výuky

Výuka je koncipována tak, aby žáky připravila na možné další studium nebo pracovní uplatnění. Jsou zařazena atraktivní témata (tvorba hry) i témata z praxe (ERP systém). Ve výuce nejsou využity jen tradiční metody (výklad, samostatné psaní programů), ale i aktivizační metody jako diskuze, týmová práce, pohybové znázornění algoritmů, opravy a optimalizace programů. Předpokládá se vedení výuky odborníkem, který má pracovní zkušenost z oblasti programování.

Hodnocení výsledků žáků

Klasifikace žáků vychází z klasifikačního řádu školy. Na konci každého tématu budou v hodinách zařazovány praktické ověřovací úkoly, které budou žáci řešit souběžně. Výstupem budou odladěné programy a skripty zapsané v programovacím jazyce. Znalost témat bude také ověřována ústním či písemným zkoušením nebo formou vytvořené a obhájené prezentace. Klasifikace bude vycházet nejen z výsledků zkoušení, ale bude zohledněn i přístup žáků k řešení jednotlivých úloh při procvičování učiva. Dalším prvkem hodnocení bude zejména v třetím ročníku projekt - samostatná tvorba počítačové hry. Zohledněn bude i zájem žáků o oblast nad rámec výuky (odborná činnost, soutěže, případná vlastní tvorba...). Hodnocení bude mít motivační charakter, žáci budou vedeni tak, aby cítili potřebu vzdělávat se s ohledem na využitelnost získaných znalostí a dovedností v dalším studiu i v praktickém životě.

Přínos předmětu k rozvoji klíčových kompetencí

Vzdělávání v oblasti programování přispívá k rozvoji těchto klíčových a občanských kompetencí:

Kompetence k učení

– umět efektivně vyhledávat a zpracovávat informace;

– využívat ke svému učení různé informační zdroje;

– znát možnosti svého dalšího vzdělávání a uplatnění, zejména v oboru a povolání.

Kompetence k řešení problémů

– porozumět zadání úkolu nebo určit jádro problému, získat informace potřebné k řešení problému, navrhnout způsob řešení, popř. varianty řešení, a zdůvodnit jej, vyhodnotit a ověřit správnost zvoleného postupu a dosažené výsledky.

Komunikativní kompetence

– porozumět běžné odborné terminologii a pracovním pokynům v písemné i ústní formě.

Personální a sociální kompetence

– podněcovat práci týmu vlastními návrhy na zlepšení práce a řešení úkolů, nezaujatě zvažovat návrhy druhých.

Kompetence k pracovnímu uplatnění a podnikatelským aktivitám

– mít přehled o možnostech uplatnění na trhu práce v daném oboru; cílevědomě a zodpovědně rozhodovat o své budoucí profesní a vzdělávací dráze;

– mít reálnou představu o pracovních, platových a jiných podmínkách v oboru a o požadavcích zaměstnavatelů na pracovníky a umět je srovnávat se svými představami a předpoklady.

Kompetence využívat prostředky informačních a komunikačních technologií a pracovat s informacemi

– učit se používat nové aplikace;

– získávat informace z otevřených zdrojů, zejména pak s využitím celosvětové sítě Internet;

– pracovat s informacemi z různých zdrojů nesenými na různých médiích (tištěných, elektronických, audiovizuálních), a to i s využitím prostředků informačních a komunikačních technologií;

– uvědomovat si nutnost posuzovat rozdílnou věrohodnost různých informačních zdrojů a kriticky přistupovat k získaným informacím, být mediálně gramotní.

Odborné kompetence

Programovat a vyvíjet uživatelská, databázová a webová řešení, tzn., aby absolventi:

– algoritmizovali úlohy a tvořili aplikace v některém vývojovém prostředí;

– realizovali databázová řešení.

Usilovat o nejvyšší kvalitu své práce, výrobků nebo služeb, tzn., aby absolventi:

– chápali kvalitu jako významný nástroj konkurenceschopnosti a dobrého jména podniku.

Mezipředmětové vztahy

Předmět navazuje na programování a vývoj aplikací. Při řešení některých problémů je nutno využívat matematických postupů. Jazykové dovednosti jsou procvičovány při tvorbě textových částí programů a dokumentací. Znalost cizích jazyků je nutná ke studiu cizí programové dokumentace, která bývá zpravidla v anglickém jazyce. K hodnocení efektivnosti některých programových řešení je nezbytná znalost principů hardware. Programování v ERP systému procvičí znalosti z ekonomických předmětů, grafické programování zase grafické a výtvarné cítění a skriptování v příkazové řádce znalosti operačních systémů. Dle

časových možností budou zařazeny úlohy procvičující další mezipředmětové vztahy (výpočet volebního výsledku, sportovní výpočty, ...).

Studijní materiály

Programování v ERP systému

http://www.delphibasics.co.uk/

https://firebirdsql.org/

Dokumentace ERP systému ABRA Gen

2D grafické programování

http://wxwidgets.org/

Abstraktní datové typy

http://www.cs.vsb.cz/benes/vyuka/upr/texty/adt/index.html

Algoritmy

https://www.algoritmy.net/

Skriptování v příkazové řádce

https://www.linuxexpres.cz/praxe/serial-o-bashi

http://www.fit.vutbr.cz/~martinek/linux/files/linuxref.pdf

https://blogs.technet.microsoft.com/technetczsk/tag/scripting/

http://geo.mff.cuni.cz/pocitace/lh-shell-scripting.pdf

Rozpis učiva a výsledků vzdělávání

Seminář z programování - 3. ročník

Výsledky vzdělávání	Tematické celky	Hod. dotace	PT
Žák – chápe, co je ERP systém a jak se používá – zná běžné ERP systémy – zná datový model ERP systému a umí jej upravovat – umí přizpůsobovat ERP systém požadavkům uživatelů – umí vytvářet programy v ERP systému – orientuje se v událostech ERP systému – je schopen vytvářet formuláře v ERP systému	**1. Programování v ERP systému** – základní pojmy – přehled ERP systémů – datový model ERP systému – přizpůsobení ERP systému – programovací model ERP systému – události objektu – události agendy – tvorba formulářů	**33**	**Ikt1**
Žák – orientuje se v nástrojích pro grafické programování – umí vytvářet formuláře – zná princip rasterizace objektů – umí využívat události k řízení běhu programu – je schopen vhodně reprezentovat objekty v 2D programu – umí realizovat pohyb 2D objektů – je schopen ukládat data programu do souboru ve vhodném formátu	**2. 2D grafické programování (jednoduché hry)** – nástroje pro grafické programování – tvorba formulářů – rasterizace objektů (úsečka, kružnice) – událostmi řízené programování – reprezentace objektů – modelování pohybu – ukládání dat programu	**33**	**Ikt1**

Rozpis učiva a výsledků vzdělávání

Seminář z programování - 4. ročník

Výsledky vzdělávání	Tematické celky	Hod. dotace	PT
Žák – chápe pojmu abstraktní datový typ – zná jednotlivé ADT – je schopen vytvářet algoritmy obsluhující ADT – umí využívat ADT ve vlastních programech	**1. Abstraktní datové typy (ADT)** – ADT seznam – ADT zásobník – ADT fronta – ADT pole – ADT množina – ADT strom	**19**	**Ikt1**
Žák – orientuje se v algoritmech řazení a vyhledávání, zná jejich vlastnosti a chápe jejich princip – dovede zvolit vhodný algoritmus a využít jej v programu – je schopen řadit daným algoritmem – zvládne daný algoritmus naprogramovat – umí určit a porovnat složitost algoritmů	**2. Významné algoritmy** – algoritmy řazení – algoritmy vyhledávání v textu – konečný automat – složitost algoritmů	**19**	**Ikt1**
Žák – orientuje se v příkazech daného operačního systému – umí vytvářet jednoduché skripty v daném operačním systému	**3. Skriptování v příkazové řádce** – příkazy ve Windows – skripty ve Windows – příkazy v UNIX-like systémech – skripty v UNIX-like systémech	**19**	**Ikt1**

Materiální zajištění předmětu

- ERP systém ABRA Gen včetně licence na skriptování,
- bezplatný operační systém Ubuntu,
- bezplatné překladače a vývojová prostředí (minimálně C/C++, Java, wxWidgets).

PŘÍLOHA P II: METODICKÝ LIST PRO POHYBOVOU VÝUKU ALGORITMŮ

Příklad 1: Výuka řadících algoritmů

Postup: Pokud vyučující chce demonstrovat složitější algoritmus, je vhodné si jej předem vyzkoušet nanečisto na náhodných příkladech, aby při předvádění před žáky netápal.

Vyučující si vybere skupinu žáků. Velikost skupiny volí s ohledem na demonstrovaný algoritmus. U jednoduchých algoritmů stačí menší počet žáků (např. 5 pro select sort), u složitějších lépe zvolit větší počet (např. 10 pro quick sort), ale zase ne příliš mnoho, aby demonstrace netrvala dlouho. Žáci se postaví do řady. Zbylé žáky lze případně využít jako ukazatele apod.

Učitel seznámí žáky se záměrem seřadit žáky dle jejich výšky. Učitel vyzve žáky, aby se pohybovali dle jeho pokynů. Učitel komentuje, výšky kterých žáků porovnává, a udílí žákům pokyny, kam přesně se mají stavět. Takto demonstruje celý algoritmus.

Možné obměny: 1. Učitel zadá žákovi, aby takto (určitým algoritmem) seřadil své spolužáky. 2. Na zemi budou nakreslena místa v počítačové paměti (řazené pole, pomocná proměnná). Každý žák musí stát v jednom paměťovém místě. Pouze jeden žák se může přemisťovat mezi dvěma jinak prázdnými paměťovými místy.

Příklad 2: Přidání/odebrání prvku ze seznamu

Postup: Vyučující si vybere skupinu žáků. Stačí poměrně malá skupinka (např. 6). Žáci se postaví do řady, kde reprezentují prvky seznamu. Jejich ruce reprezentují ukazatele na další/předchozí prvek (v případě jednosměrného seznamu je jedna ruka za zády). Jeden žák představuje hlavičku seznamu. Další žák uděluje pokyny, jak mají žáci postupovat, aby přiřadili/odejmuli nový prvek ze seznamu. Celá akce musí proběhnout efektivně. Nesmí dojít ke ztrátě ukazatele na data.

PŘÍLOHA P III: DOTAZNÍK PRO ŽÁKY (VČETNĚ SHRNUTÍ ODPOVĚDÍ)

1. **V nedávné době bylo možné si zvolit volitelný předmět. Který volitelný předmět jste si zvolil(a)?**

Volba odpovědi	Skupina 1 Celkem	Podíl	Skupina 2 Celkem	Podíl	Obě skupiny Celkem	Podíl
Seminář z programování	4	30,77%	1	7,14%	5	18,52%
Jiný z oblasti IT	6	46,15%	8	57,14%	14	51,85%
Jiný	3	23,08%	4	28,57%	7	25,93%
Bez odpovědi	0	0,00%	1	7,14%	1	3,70%

2. **Ať už jste si vybral(a) jakýkoliv předmět, jaká pozitiva vidíte na předmětu Seminář z programování (pro sebe osobně, pokud byste předmět dělal(a))?**

Kategorie odpovědi	Celkem	Podíl
zlepšení se v programování	12	44,44%
práce	4	14,81%
využití obecně	2	7,41%
různé jazyky	1	3,70%
více znalostí	1	3,70%
pokračování studia	1	3,70%
IT obor mě baví	1	3,70%
vyučující	1	3,70%

3. **A jaká vidíte negativa?**

Kategorie odpovědi	Celkem	Podíl
složitost, náročnost	5	18,52%
nebaví	1	3,70%
různé jazyky	1	3,70%
„neumím programovat“	1	3,70%
„vyučující je pohodář“	1	3,70%

4. **Za jak přínosné považujete jednotlivé tematické okruhy předmětu? Přidělte hodnocení 1 – 5, 1 znamená nejpřínosnější, 5 znamená nejméně přínosné. Jde o hodnocení, ne o pořadí.**

Tematický okruh	Skupina 1	Skupina 2	Obě skupiny Průměr

Programování v ERP systému	3,00	2,86	2,93
2D grafické programování (jednoduché hry)	2,15	2,43	2,30
Abstraktní datové typy	3,08	3,14	3,11
Významné algoritmy	2,92	2,57	2,74
Skriptování v příkazové řádce	3,00	2,57	2,78

5. Jaká témata byste si přál(a) vy osobně v takovém předmětu?

Kategorie odpovědi	Celkem	Podíl
Hry	4	14,81%
Java	1	3,70%
„jiný program než jen ABRA Gen, např. HELIOS“	1	3,70%
„co dnešní programování přináší (př. C++,Java,….)“	1	3,70%
Páv simulator 2k17	1	3,70%
„Pohodu“	1	3,70%
Optimalizace Windows	1	3,70%

6. Přináší podle vás něco využití nástroje Virtual Programming Lab (VPL) při výuce programování?

	Skupina 1		Skupina 2		Obě skupiny	
Volba odpovědi	Celkem	Podíl	Celkem	Podíl	Celkem	Podíl
Rozhodně ano	0	0,00%	0	0,00%	0	0,00%
Spíše ano	3	23,08%	0	0,00%	3	11,11%
Nevím	10	76,92%	13	92,86%	23	85,19%
Spíše ne	0	0,00%	1	7,14%	1	3,70%
Rozhodně ne	0	0,00%	0	0,00%	0	0,00%

7. Pokud ano, dovedete popsat svými slovy, co?

Bez konkrétní odpovědi, pouze odpovědi typu „nevím“.

8. Ve výuce jste viděl(a) znázornění řadících algoritmů k seřazení žáků podle velikosti. Tento způsob výuky považujete za (možno označit více možností):

	Skupina 1		Skupina 2		Obě skupiny	
Kategorie odpovědi	Celkem	Podíl	Celkem	Podíl	Celkem	Podíl
Názorný	3	23,08%	3	21,43%	6	22,22%
Zábavný	3	23,08%	3	21,43%	6	22,22%
Přínosný	7	53,85%	9	64,29%	16	59,26%
Neužitečný	1	7,69%	0	0,00%	1	3,70%
Nudný	1	7,69%	2	14,29%	3	11,11%

Zbytečně zdržující	2	15,38%	1	7,14%	3 11,11%
Jiná vlastnost (nevím)	1	7,69%	0	0,00%	1 3,70%

9. Byla podle vás hodina s příklady s neznámými kódy, kdy jste měl(a) odhalit smysl kódu a případně kód opravit, přínosná?

Volba odpovědi	Skupina 1 Celkem	Podíl	Skupina 2 Celkem	Podíl	Obě skupiny Celkem	Podíl
Rozhodně ano	0	0,00%	1	7,14%	1	3,70%
Spíše ano	4	30,77%	6	42,86%	10	37,04%
Nevím	1	7,69%	2	14,29%	3	11,11%
Spíše ne	4	30,77%	5	35,71%	9	33,33%
Rozhodně ne	4	30,77%	0	0,00%	4	14,81%

10. Proč ano, proč ne?

Kategorie odpovědi	Celkem	Podíl
- nepochopení	7	25,93%
- zmatek v hodině	3	11,11%
+ bude se to hodit	3	11,11%
+ pochopení	2	7,41%
- nebaví	1	3,70%
+ přiučení se něčemu novému	1	3,70%
- nic to nepřineslo	1	3,70%
- příkazy, které jsme se neučili	1	3,70%
+ „abych věděl jsem chytrej“	1	3,70%

PŘÍLOHA P IV: DOTAZNÍK PRO VYUČUJÍCÍHO (ODPOVĚDI TOMÁŠE VALACHA)

1. **Jaká pozitiva vidíte na předmětu Seminář z programování (pro žáky, kteří předmět absolvují)?**

   Seminář z programování vznikl jako volitelný předmět pro studenty, kteří chtějí rozšířit své znalosti z programování nad rámec klasické výuky, proto vidím pozitiva především ve větším rozvoji technického myšlení a prohloubení znalostí a dovedností v různých programovacích jazycích.

2. **A jaká vidíte negativa?**

   Negativa nevidím téměř žádná. Snad jen ta, že učitel takového předmětu musí být opravdu odborník na programování, jelikož musí znát hned několik programovacích jazyků, aby mohl své studenty posunout dále a pomohl jim při řešení problémových situací. Tzn. vedení takového předmětu nemusí být vůbec snadné.

3. **Za jak přínosné považujete jednotlivé tematické okruhy předmětu? Přidělte hodnocení 1 – 5, 1 znamená nejpřínosnější, 5 znamená nejméně přínosné. Jde o hodnocení, ne o pořadí.**
   - Programování v ERP systému 1
   - 2D grafické programování (jednoduché hry) 2
   - Abstraktní datové typy 3
   - Významné algoritmy 2
   - Skriptování v příkazové řádce 2
4. **Přináší podle vás něco využití nástroje Virtual Programming Lab (VPL) při výuce programování?**
   - **Rozhodně ano**
   - Spíše ano
   - Nevím
   - Spíše ne
   - Rozhodně ne
5. **Pokud ano, shrňte co.**

Přináší zjednodušení a zrychlení komunikace mezi žáky a učitelem. Přínos vidím především v rychlé úpravě kódu programu a v možnostech rychlého hodnocení. V praxi by se tento modul dal využít z mého pohledu na plnění domácích úkolů.

6. **Znázornění řadících algoritmů k seřazení žáků podle velikosti - tento způsob výuky považujete za (možno označit více možností):**
    - **Názorný**
    - Zábavný
    - Přínosný
    - Neužitečný
    - Nudný
    - Zbytečně zdržující
    - Jiná vlastnost: …….
7. **Byla podle vás hodina s příklady s neznámými kódy, kdy žáci měli odhalit smysl kódu a případně kód opravit, přínosná?**
    - **Rozhodně ano**
    - Spíše ano
    - Nevím
    - Spíše ne
    - Rozhodně ne
8. **Proč ano, proč ne?**

    Rozhodně ano. Žáci se snažili pochopit smysl fungování daného programu. Jejich cílem bylo seznámit se s kódem a najít chybu. Snažili se pochopit práci někoho jiného a způsob jeho myšlení. Pokud nějaký příkaz nebo část kódu neznali, nebo mu nerozuměli vyhledali si jej na internetu, což beru jako přínos vzhledem k výuce. Nehledě na to, že v profesi programátora je pochopení cizího kódu zcela běžnou záležitostí.

PŘÍLOHA P V: OBSAH PŘILOŽENÉHO CD

CD obsahuje následující položky:

- fulltext.pdf (elektronická verze této diplomové práce)
- prilohy (složka)
    - dotazniky (složka s vyplněnými dotazníky od žáků a vyučujícího)
    - prg_it_3.mbz (záloha kurzu z Moodle)