Přírodovědecká fakulta
Faculty of Science

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Srovnání výukových programovacích jazyků s ohledem na stupeň kognitivního vývoje studentů středních škol a žáků druhého stupně základních škol

Autor diplomové práce: **Jan Turoň**

Vedoucí diplomové práce: **RNDr. Jaroslav Icha**

České Budějovice | 2015

TUROŇ, J.: *Srovnání výukových programovacích jazyků s ohledem na stupeň kognitivního vývoje studentů středních škol a žáků druhého stupně základních škol* [Comparison of Educational Programming Languages with Respect to Cognitive Level of Secondary School Students, Thesis, in Czech], Faculty of Science, The University of South Bohemia, České Budějovice, 2015.

1.1 Anotace

Výuka programování bývá zařazována na SŠ a 2.st. ZŠ jako způsob realizace některých bodů RVP. V době vydání této práce jsou tyto zachyceny pouze vágně a přínos programování z pohledu RVP lze tak posoudit pouze volně. Práce rozebírá použitelnost programovacích jazyků používaných při výuce s ohledem na kognitivní schopnosti žáků a studentů a předkládá jejich srovnání, které může sloužit jako podklad pro tvorbu ŠVP.

1.2 Annotation

Education of programming language is often included in Czech secondary education as a method of implementation of corresponding school standards. By the time of publishing this thesis, these standards are only vaguely defined, so the effect of teaching programming can't be objectively evaluated from the perspective of the standards. This thesis elaborates on the usability of educational programming languages with respect to cognitive level of secondary school students, which may be used as a founded source in defining standards of particular schools.

1.3 Poděkování

Především děkuji svému školiteli, Jaroslavu Ichovi za vstřícnost v průběhu tvorby této práce, za konzultace v průběhu práce a za inspiraci k exaktnímu uvažování, kterou mne ovlivnil již léta před touto prací.

Dále děkuji Libuši Dvořákové za konzultace v oblasti sociologie a jejímu kolegovi Jiřímu Joštovi za základy psychologie, kterými mne velmi ovlivnil.

Děkuji též paní Heleně Pavličíkové za základy filosofie, díky kterým jsem se naučil kriticky myslet v jiných paradigmatech a Václavu Nýdlovi, který mi byl vzorem lidského přístupu v přírodních vědách.

Děkuji také Marii Šmilauerové, která se velice nezištným a skromným způsobem stará o příjemné prostředí naší fakulty.

Děkuji i Liboru Dostálkovi za jeho vzorné vedení Ústavu aplikované informatiky a za pomoc v administrativních záležitostech, se kterými jsem se při svém studiu potýkal.

Děkuji Břetislavu Bakalovi za pomoc v datových sítích, dopravě a pracovních záležitostech a za to, že mi pomohl udržet si zdravý rozum.

Děkuji své rodině za obrovskou podporu při svém nelineárním průběhu studia.

Závěrem bych stručně rád poděkoval i těmto vyučujícím, kteří mne každý něčím nevšedním obohatil a bez nichž by těžko vznikla tato práce: Zora Říhová, Petra Zákostelecká, Ladislav Beránek, Miroslav Skrbek, Martina Pavelková, Jan Fesl, Ruda Vohnout.

Anonymně též děkuji mnohým učitelům mimo JČU, kteří mne hodně naučili a autorům literatury, která byla podkladem mých znalostí.

2 Obsah

2.1 Seznam obrázků

3 Metodika

Cílem práce je srovnání vybraných výukových programovacích jazyků s ohledem na stupeň kognitivního vývoje studentů středních škol a žáků 2. stupně základních škol.

Jako východisko této práce byl použit RVP ZV [1a] a RVP 18-20-M01 Informační Technologie [1b] pro SOŠ. Z těchto plánů byly použity části realizovatelné výukou programování a rozebrány nejčastější metody výuky programování na školách.

S přihlédnutím ke stupni kognitivního vývoje byla navržena a zrealizovaná výuka vybraných programovacích jazyků na 2. stupni ZŠ Šobrova a ve 3. ročníku SPŠ-VOŠ Písek. Tyto aktivity byly zdokumentovány.

Na tomto základě byly srovnány vlastnosti zkoumaných jazyků a uvedeny poznatky o jejich použití ve výuce.

4 Programování na školách

4.1 Rámec výuky programování

Současná pozice výuky programování v RVP (2015) je charakterizovaná následujícími částmi:

4.1.1 Kompetence RVP ZV realizovatelné pomocí výuky programování

RVP ZV [1a] Specifikuje následující body relevantní k výuce programování:

- **5.2 Matematika a její aplikace:** dovednost provádět operaci, algoritmické porozumění (proč je operace prováděna předloženým postupem) a významové porozumění (umět operaci propojit s reálnou situací) [1a, odst. 3]
- **5.3 Informační a komunikační technologie:** schopnosti formulovat svůj požadavek a využívat při interakci s počítačem algoritmické myšlení [1a]

Algoritmické myšlení je definováno[1] jako schopnost a dovednost nalézat efektivní algoritmy při řešení problému. Algoritmus na úrovni ZŠ lze chápat jako postup řešení úlohy. Algoritmické porozumění nechť je definováno jako osvojení základních vlastností algoritmu, zejména jednoznačnosti, obecnosti a resultativnosti, což se vymezuje vůči intuitivní definici *algoritmus = co máš udělat*, která tyto vlastnosti mnohdy postrádá.

4.1.2 Kompetence RVP IT pro střední školy

RVP IT popisuje jednak **klíčové kompetence**, kde z pohledu programování je relevantní oblast *kompetence k řešení problémů*, ale také **odborné kompetence**. Ty jsou popsány jako:

[1] Prof. Rudolf Kohoutek, CSc. pro `slovnik-cizich-slov.abz.cz`

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. [1b, kap. 3.2e]

s ohledem na odpovídající uplatnění absolventa v oblasti *vývoj uživatelských, databázových a webových řešení* [1b, kap. 4]. Vezmeme-li v úvahu současný vývoj v oblasti mikropočítačů, je možné realizovat oblast programování také v této části:

Navrhovat, sestavovat a udržovat HW, tzn. aby absolventi:

- volili vyvážená HW řešení s ohledem na jeho funkci, parametry a vhodnost pro předpokládané použití
- kompletovali a oživovali sestavy včetně periferních zařízení
- identifikovali a odstraňovali závady HW a prováděli upgrade [1b, kap. 3.2a]

obecně jako programování mikroprocesorové techniky, v závislosti na ŠVP konkrétní školy.

4.1.3 Národní program vzdělávání

Národní program vzdělávání (NPV, též Bílá kniha), z něhož výšezmíněné RVP vychází, se vymezuje jako (s.6) *systémový projekt, formulující myšlenková východiska, obecné záměry a rozvojové programy, které* ***mají být směrodatné*** *pro vývoj vzdělávací soustavy ve střednědobém horizontu. Má se stát* ***závazným základem****, z něhož budou vycházet konkrétní realizační plány rezortu…*

Přičemž jeho zdroje nejsou akademické studie formulující hypotézy ověřované výzkumem (aby mohla být průběžně nezávisle ověřována jejich platnost), ale nezávazné konference, výroční zprávy a úvahy (viz prameny tamtéž). Prameny si ale žádné nároky na závaznost nečiní:

- A Memorandum on Lifelong Learning: ***Its purpose is to launch*** *a European-wide* ***debate*** *on a comprehensive strategy…*

- Vzdělávání pro 21. století: *This book is offered in the **hope that it will help in some measure in the ongoing debate** and thinking in the region on the reform and reorientation of education.*
- …atd.

V NPV není jednoznačně doloženo, o jaký výzkum se tyto vize opírají a čím je garantována jejich platnost.

4.1.4 Posun v úloze počítačů a systém CS Velké Británie

Před lety nebylo ovládání počítače pomocí periferií takto intuitivní, počítače byly především výpočetní, nikoliv komunikační zařízení. Tuto změnu reflektuje např. vzdělávací systém Velké Británie předchodem od ICT k CS roku 2013. Podobně jako u nás i zde jsou sporné body, například v Klíčové fázi 1 (Key stage 1) pro věk 5-7 let je v osnovách bod

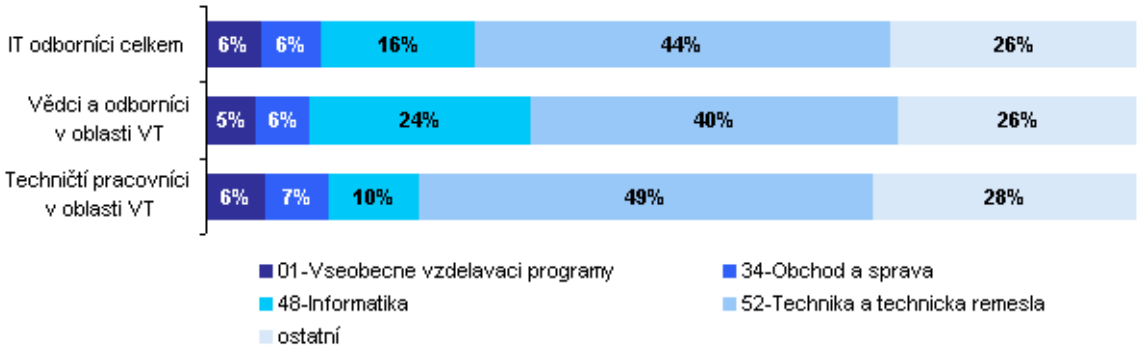
create and debug simple programs

Debug (odladění) neznamená *najděte chybu v programu* (což by snad šlo u 5-7letých dětí požadovat), ale *porozumění, kdy a za jakých okolností se program nachází v určitém stavu*, protože program, který vrací správný výsledek, může stále obsahovat závažné chyby, které se neprojevují vždy. Odladění i jednoduchého programu je náročná činnost i pro středoškolské studenty a vyžaduje předchozí znalosti a zkušenosti, například výše zmíněnou diskrepanci mezi úspěšným spuštěním a funkčností programu. Požadovanou úroveň abstrakce mají až dospívající [2, str. 372, 379]. Přiměřenější náhrada by snad mohla být *Analyze simple programs*.

4.1.5 Současné výsledky IT vzdělávání

Ze své stávající čtyřleté praxe, konzultací s kolegy i z výzkumů mé závěrečné práce na Ostravské univerzitě [7] plyne, že přibližně 10-20% studentů má vnitřní motivaci k samovzdělávání v oboru programování, dokáží získat kompetence i bez přispění školy.

Podle ČSÚ [3] má zhruba polovina IT odborníků vysokoškolské vzdělání. Z toho ale pouze 16 %(!) má vystudovanou informatiku, mnohem větším přínosem je technika. [7][8]



Obrázek 1: IT odborníci podle vzdělání dle ČSÚ, 2010

Z uvedených statistik plyne, že technika, kde se člověk setkává s VT doprovodně, je 2x (v případě vědeckých pracovníků) až 5x (v případě techniků) úspěšnější než školy, které se na IT specializují.

4.2 Implementace výuky programování

4.2.1 Efektivita výuky

RVP ZV definuje pouze výstupy. K objektivnímu posouzení efektivity naplňováním indikátorů je ale zapotřebí porovnat je se vstupy. Například IKT obsahuje bod *Základy práce s počítačem*, kde se mají žáci učit mj. používání periferií. Počítače jsou dnes ale díky zařízením s dotykovým displejem rodičů (tablety, mobily, wearables) příležitostně přístupné dětem od narození. Od desátého měsíce si děti uvědomují vztah mezi prostředkem a cílem [2]: to, že stisk tlačítka vyvolá akci počítače nebo že šoupáním na dotykovém displeji se posouvají fotky často zvládají dříve než řeč.

- V testovacích třídách zvládla nadpoloviční většina žáků šesté (11 z 20) a osmé (13 z 22) třídy ZŠ bez předchozí instruktáže vytvořit zástupce na ploše.
- Většina školních dětí má profily na sociálních sítích
- Všichni ve sledovaných třídách mají alespoň příležitostný přístup k počítači mimo školu.

Tyto a mnohé další drobné indikátory ukazují, že základy práce s počítačem jsou velkou měrou získávány bezděčně mimo školní prostředí, což RVP ZV dosud nereflektuje.

Abstraktní uvažování nutné pro algoritmickou obecnost se naproti tomu rozvíjí mnohem později a není ukončeno ani adolescencí, tedy i na středních školách je nutno počítat s tímto omezením.

I když výšezmíněné body lze splnit i bez výuky programování, existují jazyky uzpůsobené pro výuku na ZŠ, které jsou dále v této práci rozebrány.

4.2.2 Algoritmizace jako přechod od konceptuální k procedurální znalosti

Konceptuální znalosti jsou důležité pro identifikaci problému, algoritmizace jako **procedurální** znalost znamená schopnosti řešení problémů, které nejsou vzorovými příklady.

Snadno testovatelná konceptuální znalost se (zejména při frontální výuce) často omezuje na **dělení** (množinový rozklad fenoménu):

Konkrétně v matematice, která je s algoritmizací v RVP ZV propojena, se toto dělení vyskytuje např.

- aritmetické funkce jsou sčítání, odčítání, násobení a dělení
- goniometrické funkce jsou sinus, kosinus, tangens, kotangens
- atd.

Tento postup je snadno testovatelný, z pohledu kognitivních funkcí trénující paměť či dosazení do vzorečku, což je vzhledem k věku přiměřené.

Stejný princip

- algoritmy dělíme na …
- algoritmus má tyto vlastnosti …

však nelze uplatnit na algoritmizaci, jak byla definovaná výše - *nalézání* řešení problémů, což je procedurální znalost. Ta je vyvinuta u žáků na 2. stupni ZŠ (fáze *konkrétních logických* operací, Piaget, 1966) [2, str. 266].

Z toho důvodu je u algoritmizace vhodné rozlišovat kompetence na

- **vstupní**, definující pojmy k uchopení probíraného, na ZŠ vzniklé *analýzou* předchozích zkušeností žáků
- **výstupní**, vzniklé *syntézou* a procvičením vstupních kompetencí

Z pohledu algoritmizace nelze vynechat procedurální znalost. Schopnost řešit problémy nelze zaměňovat za schopnost zopakovat mírné modifikace cvičných úloh. Martin Krynický to na svých stránkách `www.realisticky.cz` nazývá *průtokové učení*, "není čas" na procedurální znalosti, na práci s již pochopeným, "jde se dál", většina žáků díky přemíry informací ve škole vše ignoruje a výuka se díky přeplněnému ŠVP stává ve skutečnosti masivním plýtváním času [9].

Dobře lze toto pozorovat u dětí, které hodně čtou: ty mají obecně lepší sloh a dělají méně pravopisných chyb i s menším množstvím konceptuálních znalostí. Přesto se u takzvaně "ryze racionální" vědy jako je matematika postupuje od teorie k praxi.

Z tohoto pohledu navrhuji následující schéma při vyhodnocování úspěchu učení algoritmizace:

1. uvedení ukázkových úloh
2. **analytické** vysvětlení základních pojmů nutných k jejich řešení (nesprávně, jen přibližně s využitím toho, co žáci znají) *konceptuální znalost*
3. řešení příkladů se stejným postupem *procedurální znalost*
4. **syntetické** vysvětlení teorie (a exaktní definice základních pojmů) na základě vyřešených příkladů *konceptuální znalost*
5. řešení problémů, kde se uplatňují teoretické poznatky *procedurální znalost*

Bod 1 je motivační, 2-3 shrnuje vstupní kompetence, 4-5 produkuje výstupní kompetence.

Tohoto postupu se drží např. učebnice na zmíněných stránkách realisticky.cz. Z pohledu algoritmizace je vhodnější, kroky 1-3 obvykle používají žáci, když si něco vysvětlují navzájem.

Výše zmíněné algoritmické myšlení a porozumění, akcentované v RVP ZV, koresponduje s opouštěním prelogického myšlení (od 6.-7. roku), nicméně až do konce základní školy je myšlení vázáno na realitu vycházející z vlastní zkušenosti. Jeho kvality jsou charakterizovány *decentrací*, *konzervací*, *porozuměním vztahů*, *reverzibilitou* a *reciprocitou* vedoucí k ***deduktivnímu*** uvažování (pochopení

významu z analýzy nových pojmů) a spolu s osvojením *kauzality*, *ohraničenosti* a *uspořádání* také k ***induktivnímu*** uvažování (syntéza zkušeností do nových teorií) [2, str. 266].

4.2.3 Object first?

OOP je dnes standardem vývoje aplikací, mající své normy UML modelování, ustálené návrhové vzory a architektonické vzory, je to desetiletí praxe umožňující psát aplikace bez nutnosti rozsáhlých (tedy nákladných) refaktorizací v průběhu vývoje.

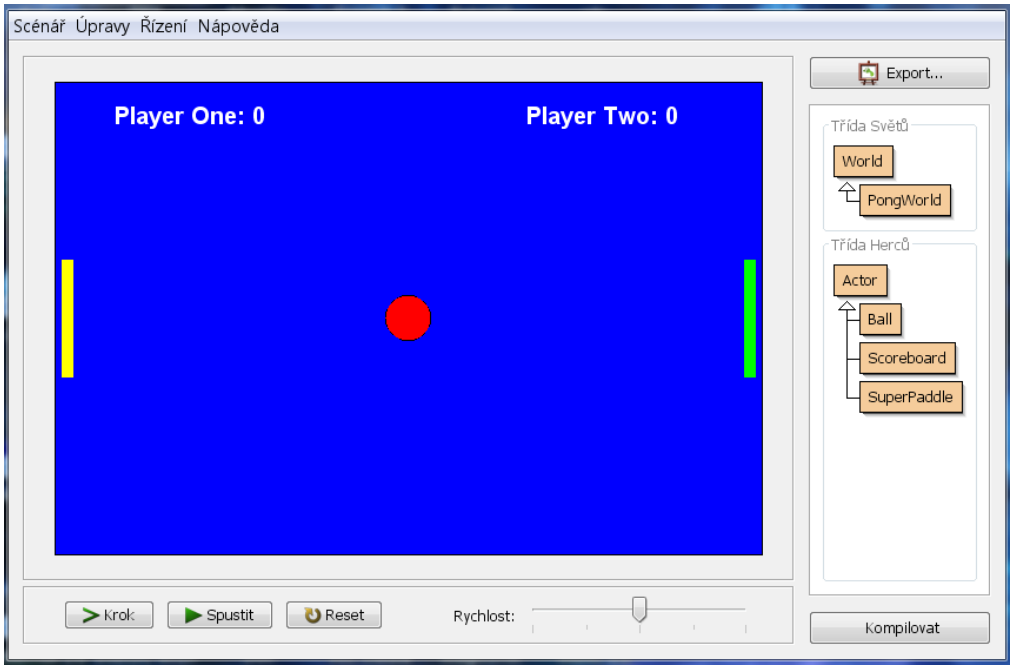
Historicky toto paradigma nahradilo strukturované programování a syntaxe nejpoužívanějších jazyků prozrazuje tento vývoj. Michael Kölling ve své knize *Object First with Java* [6] zmiňuje úskalí, které objektové programování požaduje:

- znát pojem třída
- znát význam metody main
- chápat operátor new
- přiřazení do proměnné
- deklarace proměnné vč. typu
- volání metody tečkovou konvencí

Tento (neúplný) výčet znalostí nutných pro vývoj triviálních aplikací by měl kodér znát, a pro začínající je to odrazující množství informací.

Jednou z cest je proto (zpočátku) považovat metodu main za magické zaklínadlo spouštějící program. Tento statický přístup však učí strukturované (iterativní) paradigma, a žáci se pak musí přeučovat.

Druhá možnost (Object First) je použití frameworku, např. panem Köllingem navržený Greenfoot či (od téhož autora) BlueJ, který je součástí Greenfoot IDE.



Obrázek 2: GreenFoot IDE

U nás je propagátorem tohoto způsobu např. Rudolf Pecinovský, který dle mého mínění jako jediný dobře přeložil návrhové vzory do češtiny a obohatil je praktickými (a přitom mírně pokročilým přístupnými) poznámkami.

Je však nutno si ujasnit CO chceme učit. Programování je dnes příliš široký pojem, který je žádoucí rozdělit, podobně jako bylo koncem 90. let žádoucí rozdělit předmět *Informační a výpočetní technika*. Programování (v rámci realizace ŠVP) lze předmětově dělit na *vývoj aplikací* a *automatizaci*. Rozdělení tohoto paradigmatu je možné očekávat také jako důsledky následujících diskuzí.

4.2.4 Informatické myšlení (Computational Thinking)

Tento zatím neautoritativně definovaný pojem je vymezen prvním ze zmiňovaných paradigmat, informační a výpočetní technikou. Lessner[2] shrnuje následující původ tohoto pojmu:

[2] http://ksvi.mff.cuni.cz/~lessner/w/data/_uploaded/file/papers/2014_02_lessner_didactig.pdf

- S. Pappert, tvůrce jazyka Logo, první neurčité použití pojmu CT v článku o možnostech využití počítačů při výuce matematiky
- J. Wing svou definicí *Computational Thinking is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent [computers and humans]*[3], což, jak vzápětí neformálně dodává, znamená *připuštění výpočetního řešení problému*, což lze s odkazem na Papperta volně přeložit jako *procedurální znalost algoritmizace* (výpočetní řešení implikuje algoritmické řešení)
- International Society for Technology and Education, pak charakterizují CT následovně: *Formulovat problémy způsobem, který umožňuje jejich strojové řešení, logicky uspořádávat a zkoumat data, reprezentovat data prostřednictvím abstrakcí, jako jsou modely a simulace, automatizovat řešení pomocí algoritmického myšlení (jako posloupnost kroků), odhalit, prozkoumat a provést možná řešení s cílem odhalit nejúčinnější kombinaci činností a zdrojů, zobecňovat a přenášet tento postup řešení problémů do nejrůznějších dalších oblastí.* (Lessner)

Z pohledu učitelské praxe však vyvstávají problémy párování této definice a skutečně prováděnými činnostmi žáků. Z výsledků (zvukové záznamy) zpracovaných v kapitole 6.4 plyne, že absolventi často nemají motivaci myslet informaticky, pokud je jejich obor netechnický, mnohdy si vystačí s odhadem.

Řešení této disproporce nabízí následující paradigma:

4.2.5 Creative Computing

Toto paradigma využívá přirozené hravosti dětí: spočívá v ukázání nástrojů IDE a ponechání volné ruky v kreativitě. Cílem je přitáhnout pozornost a usměrňovat ji pouze opatrně:

[3] https://www.cs.cmu.edu/~CompThink/papers/TheLinkWing.pdf

Computer science and computing-related fields have long been introduced to young people in a way that is disconnected from their interests and values – emphasizing technical detail over creative potential[4].

Creative Computing využívá osobních zájmů dítěte k počítači, povzbuzuje tvořivost a znalost, připouští a využívá toho, že děti přistupují k počítačům jako konzumenti.

Cíle deklarovaných v informatickém myšlení pak mají přirozeně vyplynout tvorbou artefaktů výšeuvedeným způsobem.

Rozdělení na IVT a automatizaci je zde klíčové: pokud jsou děti ponechány samy sobě, začnou na počítačích hrát hry a psát si po facebooku bez jakékoliv tvůrčí činnosti; tato oblast bývá označována jako *informační a* ***komunikační*** *technologie* a je zanesena v ŠVP. Náhradou slova *komunikační* za *výpočetní* vzniká definice informatického myšlení. Neoddělitelnou částí moderních počítačů je však ona komunikace ubíjející tvořivost. Z pohledu tvořivého hraní na počítači je nezbytné děti obklopit tvůrčími podněty. Například waldorfská škola v souladu s tímto razí přístup spíše **odstranit** z jejich dosahu komunikační počítače než je vyučovat (výuka počítačů na waldorfské škole Svobodná v Písku je zařazena až do posledního ročníku). Tvořivost může do světa počítačů vnést právě automatizace z toho prostého důvodu, že pokud se automat správně nesestaví a nezapojí, není s ním žádná zábava.

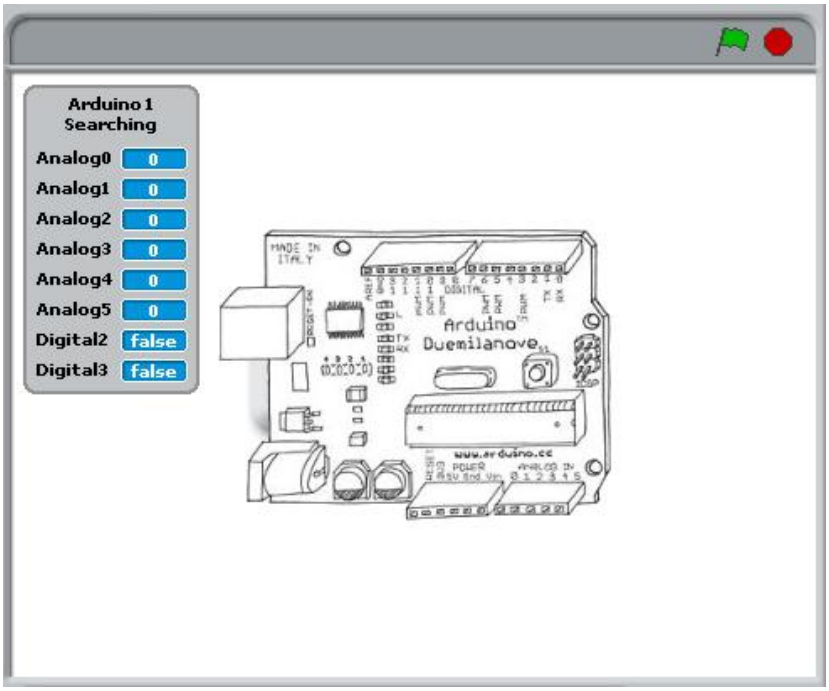
Creative computing is about empowerment. Many young people with access to computers participate as consumers, rather than designers or creators… [they] need to create the types of dynamic and interactive computational media that they enjoy in their daily lives.

4.2.6 Hardwarové vybavení pro budoucí výuku

Mikroprocesorová technika a automatizace není v RVP ZV ani vzdáleně zmiňována. Donedávna tato oblast vyžadovala technické znalosti (elektronické obvody, assembler), také mnoho komponent a vybavení vyžadující speciální učebny.

[4] http://scratched.gse.harvard.edu/guide/

V nedávné minulosti klesla cena osobního počítače z desítek tisíc i pod tři tisíce a počítač (podobně jako mobil) dnes vlastní (nebo k němu mají alespoň přístup) všechny sociální vrstvy. (Na tuto změnu dosud nereagoval RVP ZV v oblasti IT, jak je zmíněno dříve v této práci.) Ačkoliv např. slušně vybavená stavebnice Bioloid[5] vyjde cenově v řádech statisíců, její vlastnoruční tvorba není nic složitého a v současnosti vznikají levné studentské alternativy i v rámci SOČ[6]. Druhým důvodem je jazyk Scratch, který umožňuje programování pinů na desce Raspberry PI či Arduino (varianta S4A). Tímto se automatizace stává cenově i technicky přístupná základním školám pro výuku[7]. Zároveň razí Köllingem upřednostňovanou cestu IDE, které pomáhá skýt technické detaily programu, které jsou pro většinu studentů odrazující.



Obrázek 3: Arduino ve vývojovém prostředí S4A založeném na Scratch 1.4

Automatizace na nejprimitivnější úrovni však znamená dávání povelů servomotorům (či řízení pohybu při emulaci), je to sekvence činností, a zde se více hodí

[5] www.megarobot.net/www-megarobot-net/eshop/2-1-Stavebnice/3-2-Bioloid

[6] např. vítěz krajského kola SOČ pro Jihočeský kraj 2015 obor Didaktické pomůcky

[7] S ohledem k tomuto faktu byla na SPŠ-VOŠ Písek nakoupena seda mikropočítačů s plánem zařazení do výuky od roku 2017

strukturované paradigma. Objektový návrh je vhodné zařadit až při více zařízeních spolupracujících mezi sebou.

Výhoda tohoto odvětví je, že aplikace automatizace nejsou virtuální a že jsou pro děti zábavné už od prvních kroků. Po sestavení několika robotů by bylo didakticky vhodné navázat počítačovými aplikacemi na jejich ovládání, pro což by už žáci měli základy zažité. Automatizace na ZŠ není tématem této práce, pro obojí však lze použít (minimálně) jeden jazyk – Scratch, jež je v závěru této práce srovnán s ostatními jazyky z pohledu vhodnosti pro stupně kognitivního vývoje žáků a studentů.

4.3 Poznámky k RVP IT

4.3.1 Odborné kompetence: uživatelské, webové a databázové aplikace

V současnosti jsou nejpopulárnější následující řešení uvedených odborných kompetencí:

ASP

+ provázanost s .NET frameworkem, výhoda nasazení na Windows serverech

- malý podíl na trhu, vyšší náklady spojené s vývojem

PHP

+ široká komunita, mnoho rozšíření, dlouholetý stabilní kód

- slabá komunikace mezi klientem a serverem

node.js

+ rychlost, efektivita, stejný jazyk na klientu i serveru

- složitější vývoj, (zatím) slabá podpora

Java

+ široká komunita, mnoho použití (mobilní zařízení, Tomcat server, Desktop JWM, Applety)

- těžkopádnost GUI

4.3.2 Databáze

Co se týká databázových řešení, mezi nejrozšířenější a na SŠ schůdné varianty patří

- MySQL: nejčastější jako součást LAMP či WAMP, nejběžnější dialekt
- MSSQL: možné jako součást výuky na Windows serverech ADO.NET
- Access: totéž, možné zařadit do výuky kancelářských aplikací, méně časté
- SQLite: nemá SŘBD, pouze knihovna, snadné univerzální použití (autor práce je tvůrce české lokalizace webového správce phpliteadmin)

Případně další možnosti, zejména pro nadané žáky (Oracle, PostgreSQL, db4o, NoSQL řešení …), kompromisem co do rozšíření a schůdností výuky různě pokročilých studentů se jeví jako nejschůdnější varianta s MySQL či SQLite a výuka ODBC rozhraní.

4.3.3 Webový klient

Mnoho učitelů dodnes učí HTML4 a CSS2. Změna oproti HTML5 a CSS3 nejsou však pouze drobná moderní rozšíření, ale zásadní změna koncepce:

- HTML4 definuje z velké části zobrazení, HTML5 především význam obsahu a multimediální obsah bez *embed* objektů
- CSS2 řeší vzhled, CSS3 také běžné animace uživatelských akcí

Jak HTML5 tak CSS3 plní roli toho, co se kdysi označovalo jako DHTML a co bylo kdysi úlohou Javascriptu; jeho dnešní úlohou je především komunikace se serverem (AJAX či websocket).

Nereflektování tohoto vývoje je vidět například v projektu EU Peníze školám CZ.1.07/1.4.00/21.2575, prezentovaném na stránkách `zshorakhk.cz` - jakkoliv je kvalitně po odborné i didaktické stránce zpracovaný (prošel odbornou oponenturou), např. tvrzení z lekce 25 že *Jazyk XHTML je v podstatě novější verzí HTML* a že by se měl před HTML upřednostňovat platilo přibližně do roku 2004 (srovnej historický výklad vývoje W3C konsorcia[8]). Podobné materiály nelze stavět na zelené louce

[8] `www.w3.org/TR/2014/REC-html5-20141028/introduction.html#history-0`

s představou, že budou platit i za pět let: z něčeho vychází a průběžně je zapotřebí je neustále aktualizovat.

Při vývoji webového klienta není dnes už možné přehlížet mobilní zařízení, přinejmenším ustálené ovládání: akce šoupání, otočení displeje a s tím související responzivní design; tvorba mobilních aplikací je v době psaní této práce v období rozvoje plného radikálních změn: v případě zařazení do výuky je nutno počítat s tím, že během pár let budou mobilní aplikace vypadat jinak.

4.3.4 HW

Případné HW programování vyžaduje výuku jazyka C (např SDCC kompiler) či Assembleru, který má co do nasazení máloco společného s výše zmíněnými body RVP, uvedené body se vztahují spíše k základnímu servisu HW. Na SPŠ-VOŠ Písek se uvedené jazyky učí ve specializaci oboru elektrotechnika.

RVP byl psán v době, kdy mezi osobními počítači převažovaly desktopové stanice, které jsou v současnosti na ústupu. Přibývají počítače s minimem uživatelské servisní části nebo zcela bez ní: tablety, mobilní telefony, tencí klienti, *wearables*. O to více se rozšiřují datové sítě, zejména LAN.

4.3.5 Algoritmizace

Jak bylo zmíněno výše, u studentů středních škol je možné počítat s vyvinutou abstrakcí, která však teprve přechází do fáze formálních logických operací. Řešené problémy tedy mohou být rozsáhlejší, nevyžadují názorné IDE, stále by však měly vycházet z osobní zkušenosti.

U talentovaných studentů (dle zkušenosti autora 10-20%) u nichž je toto myšlení již plně rozvinuté, často dochází k divergenci jejich kognitivních schopností [2, str. 383] a získávají náskok nad ostatními studenty a v některých oblastech i nad učitelem (jelikož odborný záběr v této oblasti je široký), který se zde stává spíše mentorem, případně spolupracovníkem. Takto nadaní studenti jsou často jedinci s převahou analytických schopností, jejich praktická, sociální inteligence nemusí být na stejné úrovni [2, str. 382], vhodným doplňkem výuky je pro ně prezentace či výklad probíraného pro ostatní studenty.

4.4 Zkoumané jazyky

V rámci výšezmíněných RVP s přihlédnutím k uvedeném vývojových dispozic byly zrealizovány následující projekty:

- Test procedurálně matematických dovedností pro stanovení rozdílu mezi deklarovanými a skutečnými kompetencemi absolventů středních škol
- Projekt ZŠ – kroužek programování na ZŠ Šobrova v Písku, výuka jazyka Scratch s poznatky o jiných jazycích realizované dříve na ZŠ Tylova v Písku
- Projekt SŠ – projektová výuka Programování a vývoj aplikací na SPŠ-VOŠ Písek dle RVP IT; projekt tvorba webového serveru
- Projekt VOŠ – bloková výuka Zpracování informací: data mining a prezentace vytěžených dat v klient-server aplikacích

Na základě výsledků těchto činností je vyvozen závěr o vhodnosti vybraných programovacích jazyků a nástin diagnostiky výsledků.

	Scratch	**Lamarr**	**Web**[9]	**C**	**C#**
ZŠ	•	•	•		
SŠ	•	•	•	•	
VOŠ			•		•

[9] HTML + CSS + Javascript + PHP

5 Výzkumná část

5.1 ZŠ

Původní plán založit kroužek nebyl realizován. Jeho plánování s několikaměsíčním předstihem na dvou nejmenovaných základních školách se setkalo s následujícími obtížemi:

- administrativně legislativní (vytvoření zaměstnaneckého vztahu, smlouvy o pronájmu počítačové učebny, vykazování)
- organizační (coby pro školu nízkoprioritní činnost je vše odkládáno a je nutno se opakovaně připomínat)
- sociální (zavedené kroužky mají tradici a přes úspěšné opakované prezentace se jim nepodařilo konkurovat a naplnit minimální kapacitu)

Výsledná skupina, která za podpory rodičů fungovala půl roku v počtu šesti (později pěti) žáků šesté až osmé třídy v prostorách divadla a čajovny nebyla statisticky významná z pohledu této práce.

Podporu jsem nalezl až na ZŠ Šobrova v Písku, kde mi bylo umožněno po dobu výzkumu převzít výuku za účasti učitelek jako asistentek.

5.1.1 Zkoumané jazyky a formy výuky

V rámci této práce byly zkoumány následující jazyky

Lamarr

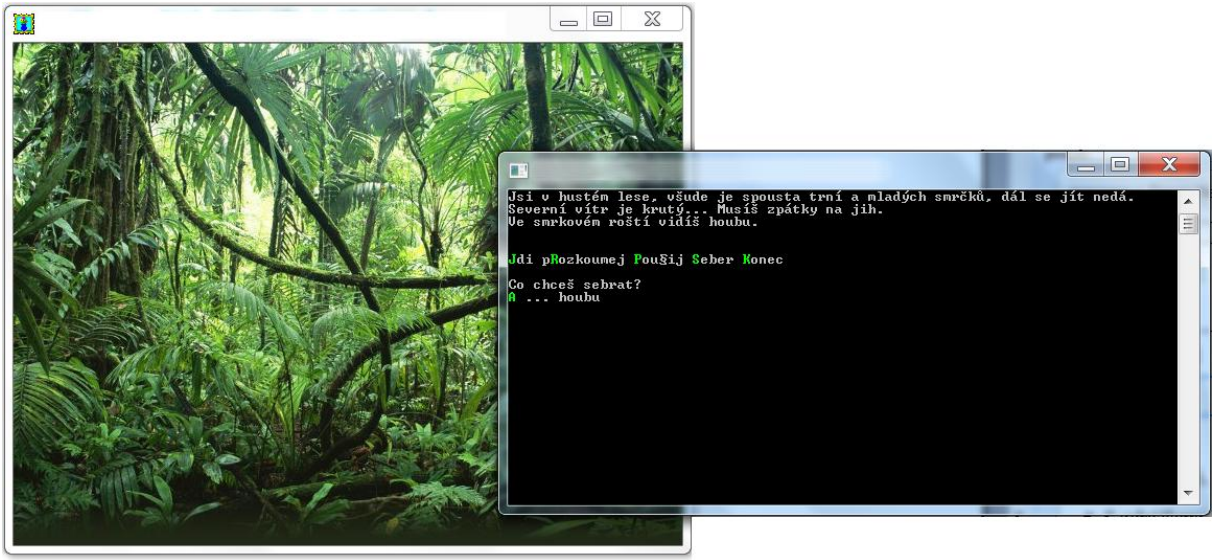
Experimentální jazyk pro tvorbu textových her bez vývojového prostředí. Byl použit v kroužku šesti studentů ZŠ 6.-9. třídy. Princip hry je tvorba příkazů pro průchod orientovaným grafem, podrobnosti a ukázky na

`https://github.com/janturon/Lamarr`

a také v [e1a]. Výhoda tohoto jazyka je možnost vytvoření hry bez znalosti podmínek, programových struktur, proměnných a dalších pojmů. Schopnější studenti mohou

použít abstraktnějších struktur jazyka umožňujícím sbírání předmětů, použití náhody, stavů a proměnných [e1b].

- Žádnému žákovi nedělalo problémy představit si průchod cca desetibodovým grafem reprezentovaným jako průchod z místnosti do místnosti. Přes vyšší nároky na představivost je to zafixovaná zkušenost [e1c].
- Procedurálně dokázal stavy a předměty použít pouze 1 žák ze šesti. Pochopení principu hotového programu s těmito rysy zvládli 3 žáci.



Obrázek 4: ukázka hry v jazyce Lamarr

HTML

Tento jazyk byl použit pouze okrajově v uvedené skupince 6 žáků, dále pak ve třetím ročníku SPŠ-VOŠ Písek spolu se srovnáním zkušeností kroužku na ZŠ Benešova v Písku.

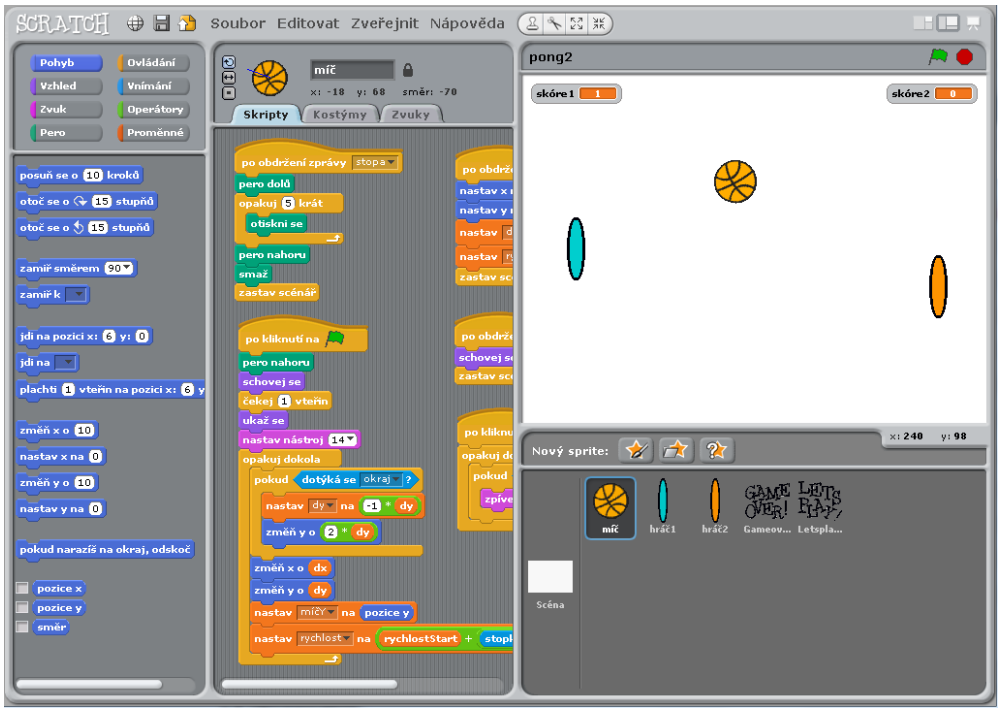
HMTL je značkovací jazyk bez proměnných, podmínek a dalších programových struktur, který ale vyžaduje procedurální znalost relativně velkého množství značek, pro tvorbu efektnějších výstupů také jazyk CSS s rozdílnou syntaxí i paradigmatem.

Žáci jsou schopní mírných modifikací hotových programů, ve většině však ne samostatné kreativní práce, kde mají v zadání vzhled webové aplikace, která není mírnou modifikací dříve vysvětlovaných. Tvorbu tabulek typu rozvrh hodin (spojování buněk, zadané ohraničení a zvýraznění obsahu) zvládlo samostatně 90% studentů SPŠ až po čtyřech hodinách opakování [e6].

Přes svou syntaktickou jednoduchost se zdá být HTML příliš obtížný pro studenty ZŠ.

Scratch

Scratch byl navržený v laboratořích MIT pro výuku od prvního stupně ZŠ. V online průvodci Scrachem[10] je doporučená metoda výuky *Creative Computing*. Tuto metodu jsem využil pouze částečně z důvodu nedělené hodiny s 25 žáky 6 a 8 třídy s využitím asistentky.



Obrázek 5: Scratch IDE

Bohaté IDE a snadnost tvorby animací rychle získala pozornost žáků, děti se na hodinu těšily. Analytická část, kdy žáci kopírovali komentovanou tvorbu učitele promítanou na projektor, probíhala bez problémů [e4b].

V syntetické části (například změna pohybu míše z horizontální na diagonální) byly mezi žáky velké rozdíly, které se nepodařilo zcela eliminovat:

- někteří žáci byli zvyklí z předchozích hodin, že jakmile se neprobírá, mohou si hrát hry, někteří byli dokonce zvyklí nosit sluchátka
- tiší žáci nedostávají ve velké skupině tolik prostoru jako temperamentnější

[10] http://scratched.gse.harvard.edu/guide/

- talentovaní žáci, kteří zvládli zadání v prvních minutách, byli odsouzeni k pomáhání pomalejším žákům; talentovaným jsem se věnoval mimo hodinu, byli schopni pokračovat mnohem rychlejším tempem
- pomalí žáci výrazně zpomalují tempo tvorby hry, někteří spolužáci s nimi ztrácejí trpělivost

Obrázek 6: ukázka žákovské tvořivosti [e4c]

Problémy nastaly zejména v závěrečné části, kdy odraz míče vyžadoval práci s dvěmi proměnnými. Část žáků toto nepochopila, udržení pořádku ve třídě vyžadovalo výchovné zásahy, což bylo na úkor žáků, které to zajímalo a jejichž iniciativní dotazy jsem byl nucen řešit mimo výuku po sociálních sítích.

5.1.2 Forma výuky

Frontální výuka informatiky, která obvykle probíhá s využitím projektoru, s uspořádáním třídy, kde žáci sedí zády či bokem k učiteli, aby měl tento přehled o jejich displejích, má dle mého pozorování následující nedostatky:

- žáci při výkladu převážně mechanicky bez přemýšlení [9] kopírují činnost na projektoru
- žáci se naučí jen mírné modifikace předváděných úloh, bez kreativního myšlení, které je implicitně požadováno v RVP. Aby výuka byla plynulá, má učitel na zpracování případných dotazů pouze jednotky sekund – v takto vymezeném času lze vyřešit pouze triviální dotazy (syntaktická chyba, kam kliknout apod.), zatímco kreativní dotazy (jak vyřešit problém za jiných podmínek, jestli je schůdný jiný postup) vyžadují více času a často nebývá

prostor využít tuto pedagogickou situaci jako součást výkladu pro ostatní z důvodu přílišného množství detailů vyžadující hlubší a delší koncentraci, než bývá většina třídy schopna.

Výuka s asistentkou má výhodu paralelních činností. Učitel si ve fázi výkladu nové látky vybírá pouze ty dotazy, které rozvíjejí probírané téma, což činí výklad pro většinu plynulý. Ostatní dotazy mající malý význam pro ostatní by samotný učitel byl nucen přeskočit nebo odložit na individuální vysvětlení po hodině. Asistentka se mohla soustředit na tyto dotazy a drobné jevy ve třídě (jako problém s přihlášením) a pomáhat tak ostatním neztratit nit. Spojená třída se dvěma učiteli je zváženíhodná alternativa k dělené třídě.

5.2 SŠ

Výuka na SPŠ probíhala v dělené po 15 žácích třídě 3. ročníku specializace informační technologie.

Na SŠ lze jako asistenta využít nadaného žáka, pokud má kompatibilní přirozenou sociální roli ve třídě, komunikativní osobnost a ochotu k této činnosti [7].

Pokud takový student ve třídě není, lze alespoň vyzvat studenty zvládnuvší činnost dříve k pomoci pomalejším kolegům.

U středoškolských studentů více vynikají rozdíly mezi individuálními dovednostmi ve třídě. Klement a Lavrinčík [10] pro tento případ doporučují projektovou metodu, která se též používá při výuce odborných předmětů na SPŠ-VOŠ Písek, kde také probíhal výzkum této práce.

5.2.1 Zkoumané jazyky a formy výuky

Scratch

V rámci Dne otevřených dveří na SPŠ byla pro zájemce připravena soutěž, kde měli žáci sestavit hru Pong na základě ukázkových částečných řešení a intuitivně pochopit práci v IDE [e4d]. Zúčastnilo se jí 11 lidí.

Lamarr

Jazyk Lamarr byl zkoumán pouze okrajově v rámci dvou hodin výuky, kde se studenti rozdělili do tří skupin po pěti studentech, s rozdělenými kompetencemi:

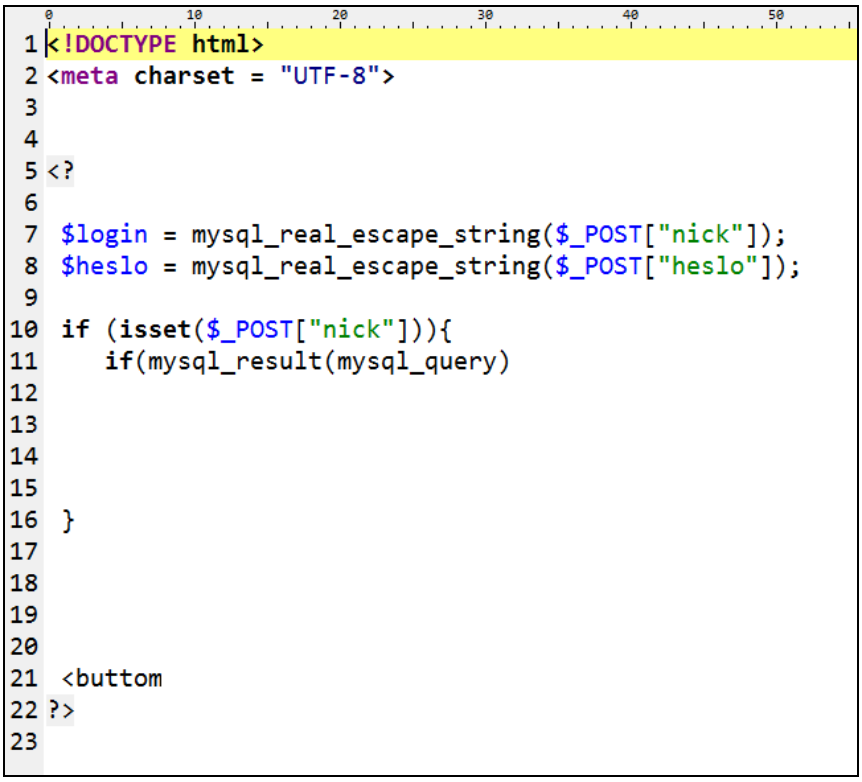
- tvorba kódu
- scénář
- grafika
- dokumentace

HTML

Výuka HTML probíhala frontálně, jelikož byla součástí ŠVP. V prvních dvou měsících byl obsahem návrh statických stránek (HTML5 + CSS3), následně tvorba klient-server aplikací (Apache + PHP + SQLite).

SQLite je vhodnější alternativa oproti sestavám EasyPHP, WAMP či XAMPP (Apache + PHP + MySQL), jelikož není zapotřebí nastavovat server (SQLite je výchozí knihovna v PHP) a pro účely výuky jsem též lokalizoval a pomohl odladit administrační rozhraní PhpLiteAdmin (www.phpliteadmin.org), studenti tak mohou vyvíjet aplikace rychleji.

Oproti žákům ZŠ je SŠ studenty obtížnější pro něco nadchnout, též jsou mezi nimi výraznější rozdíly [7]. Na jednu stranu se ve třídě obvykle najde student, který v nějaké oblasti do konce svého SŠ studia předčí v nějaké oblasti svého učitele, na druhou stranu se ve třídách vyskytují studenti, kteří od prvního ročníku nemají žádné procedurální znalosti, do vyšších ročníků postupují jen s konceptuálními znalostmi [e6]:



```
1  <!DOCTYPE html>
2  <meta charset = "UTF-8">
3
4
5  <?
6
7   $login = mysql_real_escape_string($_POST["nick"]);
8   $heslo = mysql_real_escape_string($_POST["heslo"]);
9
10  if (isset($_POST["nick"])){
11     if(mysql_result(mysql_query)
12
13
14
15
16  }
17
18
19
20
21  <buttom
22 ?>
23
```

Obrázek 7: dvouhodinová práce studenta po třech měsících výuky

C

Jazyk C byl dle starého ŠVP vyučován v druhém ročníku, po konzultaci s ostatními vyučujícími a vedením školy byla přesunuta do třetího ročníku.

V tomto jazyce potřebují studenti hlubší znalosti hardware (paměťové operace) software (knihovny, kompilátory) i další (kompilační proces, *toolchain*).

IDE těchto jazyků (Visual Studio, QT, C++ Builder, Dev-C++, Eclipse) bývají odrazující svým rozsahem a začátečníky spíše matou množstvím svých voleb. Vhodnější bývá integrace kompilátoru do textového editoru zvýrazňujícím syntaxi (PSPad, Sublime).

Ani tak se studenti druhého ročníku obvykle nedostali za ukázky příkladů z hodin, většinou ani nebyli schopni identifikovat chybu.

V přílohách je výběr z probíraných úloh [e3]:

- řešení *doublebufferingu* na OS Windows
- ovládání motoru pro desku Arduino Duemilanove
- aplikace pro měření rychlosti reakce (měření času s přesností na ms)

5.3 VOŠ

Výuka probíhala v pětihodinových blocích 1x týdně. Studenti VOŠ mají vývojově obecně vyšší schopnosti abstrakce i koncentrace než středoškoláci a s níže uvedenou strukturou bloku neměli větší problémy s udržením pozornosti.

Bylo schůdné po výkladu zadat náročnější úkol vyžadující řešení na delší dobu, během níž stihne přednášející všechny obejít a zpravidla vyřešit většinu problémů (i těch kreativních) individuálně. Vzhledem k tomu, že mezi studenty byly výrazné rozdíly a atmosféra třídy celkově tvůrčí, byla spolupráce podporovaná.

5.3.1 Zkoumané jazyky a formy výuky

C#

Výuka probíhala projektově v pětihodinových blocích v rámci předmětu Zpracování informací. Struktura bloků byla následující:

1. V úvodní fázi vyučující vysvětlil problém s nejednoznačným řešením a nástroje a postupy vedoucí k dílčím krokům řešení.
2. V pracovní fázi studenti experimentovali s uvedenými postupy s cílem získat požadovaný výsledek. Vyučující obcházel studenty a konzultoval jejich postupy, případně nastiňoval alternativy.
3. V závěrečné fázi byly postupy studentů srovnány, vysvětleny jejich přednosti a nedostatky, možnosti a další teorie vyplývající z řešení.

Tento cyklus se v průběhu pětihodinového bloku 2-3x zopakoval.

5.4 Měření matematických kompetencí

Matematika je s algoritmizací je úzce propojena (RVP ZV 5.2). Názor, že matematika je základ vzdělání, je na školách všeobecně rozšířen. Prakticky ale skutečné reformy (změny JAK učit) nepřicházejí formou metodik stanovených ministerstvem, ale jako individuální iniciativy jednotlivců (Pestalozzi, Montessori, Steiner, Rouseau a další), zatímco příkazy shora většinou končí jako další administrativní zátěž.

Ministerstvo školství často svá rozhodnutí nepodkládá fakty, ale neoponovanými závěry diskuzí odborníků (např. Národní program vzdělávání, rozebráno výše).

V rámci této práce je podstatné, jak objektivně určit pokrok či úpadek výuky, zde algoritmizace, potažmo procedurálních matematických dovedností. Byl proveden následující výzkum tvořený pěti otázkami opírající se o středoškolské matematické znalosti [t2].

- První otázka na nepřímou úměrnost obsahovala nadbytečné údaje, ukazuje také schopnost vyloučit nepodstatné údaje z výpočtu (ve školních příkladech se často uvádějí jen relevantní údaje, čímž se redukuje schopnost orientace v problémech podstatná pro algoritmizaci).
- Druhá otázka na sestavení rovnice naopak neobsahovala údaje nutné k výpočtu, řešitelé si je museli cílevědomě vyhledat.
- Třetí příklad na podobnost trojúhelníku a goniometrické funkce je úvahový: řešením není číslo, ale algoritmus či vzorec.
- Čtvrtý příklad zahrnuje základy fyziky (rovnoměrný a zrychlený pohyb) a kvadratickou rovnici.
- Pátý příklad také vyžaduje složitější a méně běžné znalosti (měrná tepelná kapacita, derivace), nemá jednoznačné řešení a byl zařazen spíše pro určení ochoty pustit se I do těžkých úloh (a rozpoznání, že jde o těžkou úlohu).

Dotazníková metoda se ukázala být málo reliabilní: do řešení se pouštěli převážně lidé, kteří si matematicky věřili, zatímco mnozí pouze odkaz sdíleli, aniž by se o řešení pokusili.

Použil jsem tedy metodu dialogu s lidmi na ulici [e5c]. Ti byli vybíráni náhodně, snažil jsem se pokrýt všechny věkové skupiny po střední škole (krom důchodců díky obecně předpokládanému poklesu jejich kognitivních funkcí a časové náročnosti při vysvětlování a dialogu) a všechny společenské vrstvy (hrubý odhad dle vzhledu), aby byl vzorek co nejreprezentativnější.

Vzhledem k metodě (převážně stojící dotazovaný na cestě) jsem požadoval pouze postup řešení, nikoliv výpočet (což z hlediska algoritmizace nemá vliv). Přestože data jsem zaznamenával zcela anonymně, musel jsem ve většině případů v úvodu překonat

obavy dotazovaných. Přesto mnozí dotazovaní ochotní k odpovědi utekli, jakmile se dozvěděli, že otázky se týkají matematiky.

5.5 Měření obecných dispozic k algoritmizaci

K určení obecných dispozic k algoritmizaci byla použita modifikovaná zkouška čtení, kde byl zvolen sci-fi text (bez vazby k osobní zkušenosti, prověřující abstrakci) [t1].

Pro splnění úlohy bylo zapotřebí v textu nalézat křížové reference cizích slov; text se tedy, stejně jako zdrojový kód, nečte shora dolů, ale "zvnějšku dovnitř", od otázky ke smysluplné definici.

Test byl prováděn na studentech středních škol (převážně z Prahy) v rámci letních táborových her. Mezi účastníky byly radikální rozdíly v době vyplňování: 4 – 36 minut. 80% účastníků ve věkovém rozpětí 16-30 let vykázalo dobré porozumění textu (min 4 body z 5).

věk	body (max 5)	čas (min)
18	5	4
18	4	6
30	5	8
21	5	9
16	2	10
18	4	11
19	3	11
26	4	14
18	4	17
21	5	17
22	4.5	18
18	1.5	20
21	4	30
16	4	36

Původně zamýšlená druhá část testu, která měla měřit schopnost pochopit algoritmus tvorby HTML tabulek, nakonec nebyla realizována. Ukázalo se, že většina účastníků již webové jazyky i algoritmizaci do jisté míry ovládá.

Účastníci oněch táborových aktivit byli přispěvatelé tématického diskuzního fóra založeného na phpBB, které značkovací jazyk používá pro vkládání příspěvků, účastníci tak byli již předem vybaveni procedurální znalostí získanou mimo výuku.

6 Seznam aktivit

6.1 ZŠ

V rámci výuky programování byly realizovány následující úkoly: seznámení s vývojovým prostředím a tvorba hry Pong [e4a].

6.1.1 Seznámení s vývojovým prostředím Scratch

lekce 1

téma: představení Scratch;

cíl: orientace v IDE Scratch, tvorba jednoduchého programu

- **úvod hodiny:** Představení jazyka Scratch, programování jako skládání dílků puzzle.
- **analýza:** Zeleným praporkem se spouští program, stopkou se zastavuje, dílky do sebe musí zapadat.
- **proceduralizace:** Napište program, kde se kocour sklouzne 2x. Zkuste použít i jiné dílky.
- **syntéza:** Program vždy začíná praporkem a končí stopkou. Máme dva různé druhy dílků: oranžové Ovládání (jak má program probíhat) a modrý Pohyb (co má kocour dělat).
- **závěr hodiny:** Stáhněte si jazyk Scratch ze scratch.mit.edu, vyzkoušejte si doma

lekce 2

téma: animace a zvuky

cíl: zaujmout, povzbudit kreativitu do dalších hodin

- **úvod hodiny:** Pochlubte se, co jste si doma zkusili
- **analýza:** Máme i tmavě modré dílky (Vzhled), se kterými jsou spojeny Kostýmy objektů. Nachové dílky (Zvuk) se dobře kombinují s dílkem Opakuj.

- **proceduralizace:** Zkuste si nakreslit nové kostýmy. Kdo nakreslí nejlepší a kdo vytvoří nejlepší hudbu?
- **závěr hodiny:** Doma si dodělejte animaci. Ukázat žákům, jak publikovat projekt na web. Žáci učiteli pošlou ID svého projektu.

lekce 3

téma: objekty a události

cíl: sestavovat program objektově.

- **analýza:** Zkuste si překliknout mezi Scénou a Kocourem: oba objekty mají svůj vlastní kód a posílají si zprávy. Do scény dáme kód pro úvodní znělku, pošleme Kocourům zprávu, aby se sklouzli, a pak ve scéně zahrajeme závěrečnou znělku.
- **proceduralizace:** Zkuste si nahradit dílek *rozešli všem kocour1-skluz a čekej* dílkem *rozešli všem kocour1-skluz*. Co se změnilo? Vložte si do animace další sprite (tlačítko *Výběr nového sprite ze souboru*), který něco udělá po obrdžení zprávy.
- **syntéza:** Program se skládá z objektů, které si mezi sebou posílají zprávy.

Žáci (i ti talentovaní) často sestavují aplikaci jako posloupnost příkazů, program se pak brzy rozroste do komplikovaných struktur nad jejich chápání. Objektový přístup tento nárůst složitosti eliminuje: každý objekt má kód, který se vztahuje jen k jeho činnostem, objekty si posílají zprávy. Mnozí[11] zdůrazňují kvalitní objektový návrh v úvodu svých publikací a jeho nedodržení je všeobecně uznávaný antivzor vývoje: God object [11].

Před lekcí 3 je možné v rámci klasifikace obětovat hodinu prezentaci a známkování projektů.

[11] GUNDERLOY, M.: Z kodéra vývojářem; PECINOVSKÝ, R.: Návrhové vzory, PALETA, P.: Co programátory ve škole neučí - a další

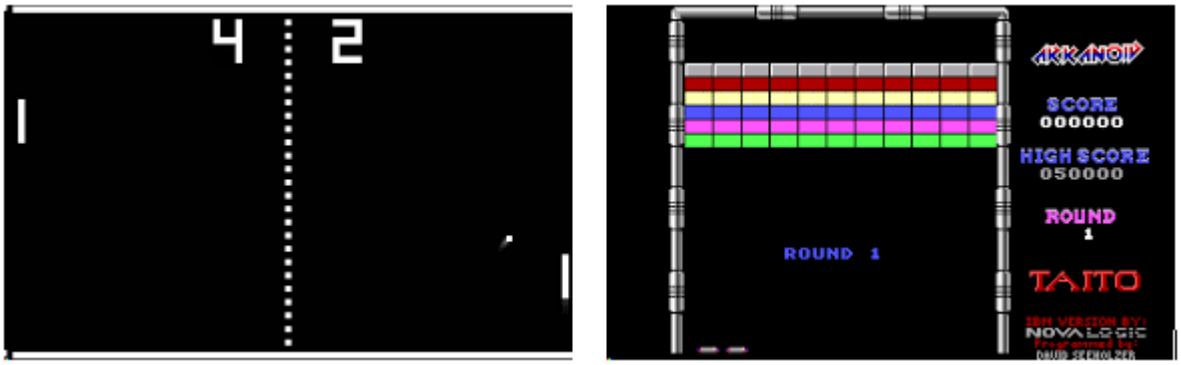
6.1.2 Tvorba hry Pong

lekce 1

téma: tvorba objektů pálek, míče a scény

cíl: chápat hru jako komunikaci mezi objekty

úvod hodiny: ukázka her snadno vytvořitelných ve Scratchi.



Obrázek 8: Pong a Arkanoid - ilustrace příkladů her pro jazyk Scratch

analýza: Máme připraveny objekty Scéna, hráč1, hráč2 a míč. Také jsme vytvořili proměnnou směrX, která udává, o kolik pixelů se míč pohne: kladné číslo je doprava, záporné doleva.

- Scéna řídí hru: po kliknutí na praporek rozešle všem objektům zprávu start
- míč se po startu posune doprostřed (x=0, y=0), nastaví směrX=3 (pohyb doprava), a poté se hýbe v zadaném směru
- hráč 2 se ovládá šipkami (podívejte se, jak je to udělané). Po startu se posune doprostřed (y=0), a poté pořád dokola testuje, jestli se nedotýká míče. Pokud ano, nastaví směrX=-3 a míč se začne pohybovat doprava (odraz)

proceduralizace:

- Udělejte podobným způsobem pohyb hráče1. Ovládejte ho klávesami q,a aby se dva hráči vešli na klávesnici.
- Rychlost hry se bude časem zvětšovat. Místo +3 a -3 nastavujte proměnnou směrX na hodnotu proměnné rychlost (vytvořte ji na cihlové kartě Proměnné), kterou po startu nastavte na hodnotu 3. Výraz -1 x rychlost uděláte pomocí dílku na zelené kartě Operátory.

- **syntéza:** Všimněte si, kdy a jak se v průběhu hry mění proměnné rychlost a směrX.

lekce 2

téma: vlastnosti objektů

cíl: schopnost rozpoznat zodpovědnost objektů

analýza: Přibyly proměnné směrY (udává směr pohybu míče nahoru a dolů; kladné číslo nahoru, záporné dolů) a proměnná míčY: když se míč srazí s pálkou hráče, odrazí se šikmo podle toho, jak moc na okraji se dotkli. Pálka tedy potřebuje znát Y pozici míče. Tu nastavuje míč po každé změně pohybu.

proceduralizace: Ve skriptu míče máte připravenu podmínku po dotyku míče s okrajem. Doplňte tam kód, aby se míč od vodorovného okraje odrážel.

syntéza: Scéna nastavuje proměnné rychlost, směrX a směrY. Ty ale popisují vlastnosti míče, měly by se proto nacházet v míči. Přesuňte je tam.

Místo abychom vytvořili proměnnou míčY, kterou používáme v pálce, jsme mohli vytvořit proměnné pálka1Y a pálka2Y a ve skriptu míče podle místa dotyku pálky určit, kam se míč odrazí. To však nedává smysl: pálka rozhoduje, kam se míč odrazí, proto je míčY správně, i když obě možnosti by byly funkční.

Rozbor vhodnosti jazyků ve srovnání se Scratchem

Co se týče přirozeného vývoje kognitivní schopnosti jsou obohaceny rychle se rozvíjejícími schopnosti abstrakce už na druhém stupni ZŠ

lekce 3

téma: dokončení hry Pong

cíl: prohloubení schopnosti rozpoznat zodpovědnost objektů

analýza: Do kódu přibyla proměnná míčX, podle které poznáme, jestli míč přeletěl pálku. Který objekt by to podle vás měl řešit?

- míč by měl kontrolovat, jestli se nachází za jednou z pálek, a pokud ano, měl by ostatním rozeslat zprávu o konci hry
- pálka by měla kontrolovat, jestli se míč nenachází za ní, a pokud ano, měla by ostatním rozeslat zprávu o ukončení hry

- Scéna by měla kontrolovat, jestli se míč nenachází za žádnou z pálek, a pokud ano, měla by ostatním rozeslat zprávu o ukončení hry

Uvědomte si, co je zodpovědností každého objektu: míč určuje pravidla odrazu, pálka pravidla odpalu a Scéna pravidla hry.

proceduralizace: V připraveném cvičení je zapsáno počítání skóre u prvního hráče. Doplňte skóre u druhého hráče[12].

poznámky k lekcím

Uvedené hodiny neměly z časových důvodů evaluační fázi: nebylo v mých kompetencích známkovat na hostující škole, vzhledem k dotaci jedné hodiny týdně na ročník (často zkrácené) nebyla příležitost zařadit evaluaci do hodiny.

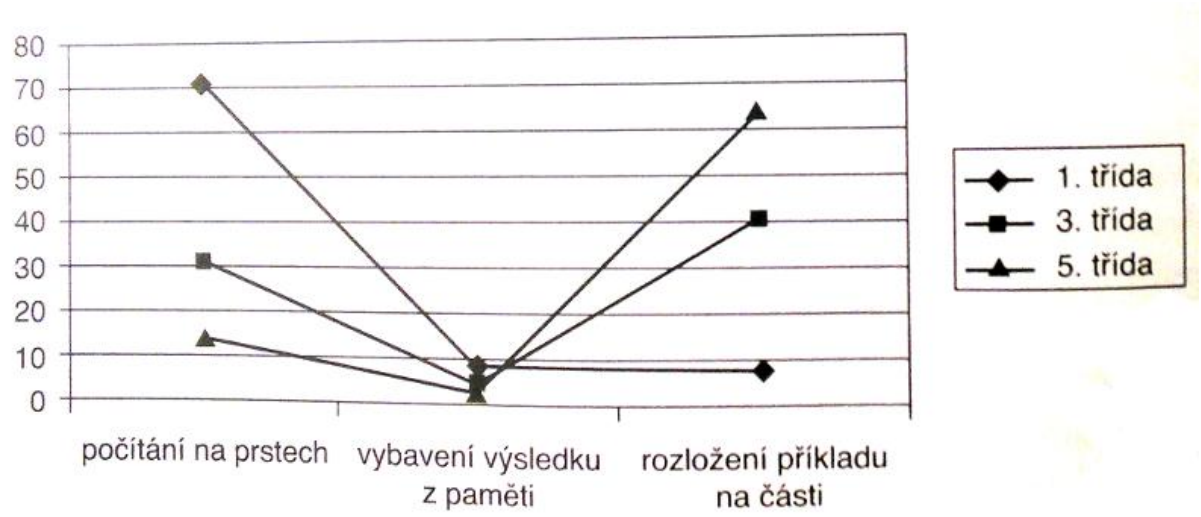
Původně plánovaná závěrečná hodina, kde by žáci prezentovali své výtvory před třídou, nakonec z technických důvodů nebyla realizována.

Hodina byla rozdělena úvodní části a dvou bloků:

- v úvodní části (cca 10-15 minut) byl zadán krátký úkol ověřující znalosti z minulé hodiny, poté byl shrnut obsah stávající hodiny
- v první části bloku (5-10 minut) byly ukazovány na projektoru nové věci s vyžadováním přesné součinnosti žáků

v druhé části bloku (5-10 minut) byl dán prostor kreativitě, tvorbě efektů a animací; zde má učitel čas řešit individuální dotazy a obejít žáky. Rychlejším žákům byl dát pokyn pomoci pomalejším sousedům, zvýšení hluku ve třídě v tomto bloku nevadí. Tento blok slouží také k občerstvení sil, jelikož žáci na ZŠ nevydrží 45 minut koncentrace.

[12] V pokusné třídě měli s tímto žáci problémy, u šikovnějších je možné umazat další část kódu skriptu Scény.



Obrázek 9: Vývojově podmíněná proměna strategií používaných při řešení složitějších příkladů (Geary, et al. 2004 in [2])

Z grafu je patrné, že s rozvojem abstrakce ji děti u složitějších úloh upřednostňují, že samy preferují dekompozici, obojí jsou základní principy algoritmizace. U starších dětí z těchto důvodů jazyk Scratch přestává být vhodným, skládání geometrických obrazců už je příliš konkrétní a spíše zdržuje.

Jako vhodný jazyk se zde jeví autorem této práce vyvinutý Lamarr. Ten je uzpůsobený k tvorbě textových her, v nejjednodušším případě mapovatelných jako průchod orientovaným grafem, přičemž způsob zápisu již nejsou geometrické obrazce, ale textový zdrojový kód. Na této úrovni je žákům srozumitelný již na druhém stupni: test proběhl na skupině dětí ZŠ Tylova v Písku, po krátkém vysvětlení byly hned schopny sestavit hru o rozsahu cca deseti uzlů (ukázky prací jsou v příloze 2). Část z nich byla schopna přímo psát kód i bez tvorby plánu hry, z hlavy. Příčinou je patrně orientace na prostorovou paměť, čehož lze využít i při diagnostice.

Na druhou stranu rozšíření hry o sběr a používání předmětů činilo problémy i studentům druhého ročníku SPŠ-VOŠ Písek: pouze jeden ze tří týmů byl schopen smysluplně zapsat použití předmětů ve hře [e1c]. Myšlení nezávislé na osobní zkušenosti (stádium formálních logických operací dle Piageta) se postupně rozvíjí až v adolescenci [2, s. 379-390] a při kódování popisu uzlu je třeba popsat všechny jeho stavy, tj. případy, kdy hráč má i nemá předmět. Tato představa více situací najednou je však za osobní zkušeností.

Na ZŠ jsou tedy žádoucí tyto vlastnosti

- názorné, netechnické, lokalizované IDE
- srozumitelný vstup a výstup, nejlépe audiovizuální
- návaznost na osobní zkušenost

6.2 SŠ

6.2.1 Scratch

- 3 zvládli hru sestavit zcela (2 měli předchozí zkušenosti s programováním v Baltíku, 1 neměl žádné předchozí zkušenosti)
- 6 zvládlo hru sestavit částečně (pohyb pálky a míčku bez vzájemné interakce)
- 2 zvládli pouze doplnění pohybu pálky

Zadání soutěže

Viz elektronickou přílohu [e4d].

Vítej v programátorské soutěži, kde v jazyce Scratch vytvoříme klasickou hru Pong: míč odrážený pálkami dvou hráčů.

V soutěži projdeš třemi levely: otevírej si postupně soubory level1.txt, level2.txt a level3.txt. Ke každému levelu je přiložen soubor s příponou sb, který si otevři v prostředí Scratch.

- Level 1 by měl zvládnout každý.
- Level 2 už vyžaduje trochu přemýšlení.
- Level 3 je pro ty nejzdatnější.

V levelech plníš úkoly, za které dostáváš body. Pokud se zasekneš, můžeš se zeptat. Vyřešení úkolu s asistentem tě ale stojí mínus jeden bod.

Pokud tě program scratch zaujal, stáhni si jej zdarma z https://scratch.mit.edu

Doma pak můžeš svůj program dotáhnout do konce. Konečná hra by mohla vypadat takto: https://scratch.mit.edu/projects/87024414/

Pokud chceš, sdílej svůj projekt na webu (položka Share v menu) a pošli odkaz na mail janturon@email.cz - nejlepší práce vychválíme na našich facebookových stránkách školy https://www.facebook.com/spsvospisek

Hodně štěstí.

Obrázek 10: Simulace perspektivy v jazyku Scratch

6.2.2 HTML a design

Výuka HTML [e2a] probíhala frontálně v souladu se ŠVP [e2b].

1. část Stručný přehled historie od HTML3.2 k HTML5, ukázky webů a jejich tvorby.

2. část Přehled základních tagů pro tvorbu jednoduchého obsahu v HTML5, práce s referenčními příručkami.

3. část Samostatná tvorba stránky: odkazy, tabulky, obrázky, rozložení

4. část Základy CSS3, použití vektorové grafiky a FontAwesome

6.2.3 PHP, klient-server aplikace

5. část HTTP komunikace (formuláře, GET, POST)

6. část práce se soubory, základy SQLite (čtení a zápis do tabulek, php_sqlite3.dll)

7. část základní útoky a zabezpečení (XSS, SQL inject, phishing)

Z 28 studentů 2. ročníku většina nabyla znalosti pouze konceptuálně (viz. Obrázek 6).

U nadaných studentů byly obsahy uvedených hodin hluboko pod jejich schopnostmi, jejich rozvoji jsem se věnoval formou mentoringu mimo výuku:

1. měsíc Demonstrace tvorby zakázky pro firmu Global Premium - editor plechovek.

2. měsíc Řešení dílčích úkolů tvorby aplikace pro monitoring metropolitní sítě firmy Cryonix.

6.3 VOŠ

Výuka probíhala v pětihodinových blocích 1x týdně. V rámci ŠVP [e6] probíhal výzkum v období října a listopadu, byla probrána následující dvě témata.

1. téma - Informační systém: správa SQLite databáze (phpLiteAdmin, webová aplikace na serveru), tvorba klienta (desktopová C# aplikace). V úvodní fázi byl popsán IS a jeho životní cyklus, základy .NET Frameworku a použití knihovny třetí strany (System.Data.SQLite.dll). V závěrečné fázi byly vysvětleny UML diagramy užití (poté, co studenti napsali první kusy vlastního kódu), refaktoring a iterace životního cyklu, metodiky návrhu IS.

2. téma - Tvorba aplikace: vytěžení dat ze serveru mesta.obce.cz a tvorba klientské aplikace nad těmito daty působící. V úvodní fázi byl vysvětlen princip činnosti webového robota (crawler, jako jednovláknová aplikace), GUI .NET Frameworku, globální architektura IS. V závěrečné fázi byly vysvětleny vybrané organizační a návrhové vzory a antivzory, časová složitost (doba běhu robota, než vytěží všechna data).

6.4 Matematické kompetence

Shrnutí výsledků (podkladová data viz [t1]):

- 1. otázka 13:11 (poměr úspěšných a neúspěšných odpovědí, částečné nezapočítány)
- 2. otázka: 4:20
- 3. otázka: 5:19
- 4. otázka: 2:21

- 5. otázka: nikdo
- lidé ochotní k vyplnění, kteří utekli po zadání: 5

Ať už jsou známky z matematiky jakékoliv, z pohledu procedurální znalosti vyplývá tento závěr: **polovina lidí ovládá trojčlenku, znalost středoškolské matematiky má 10-20% populace** (otázky 3-4). Vzhledem k obtížnosti realizace reliabilní metody je ke zpřesnění čísel třeba širšího vzorku populace, přesto je zjevné, že výsledky se radikálně liší od deklarovaných kompetencí RVP.

Výzkum byl prováděn velmi hrubě (nesledovaly se žádné ukazatele krom odpovědí na úlohy z důvodů už tak velmi nízké ochoty ke spolupráci). Má-li nějaký systém na současné úrovni poznatků výrazně více než oněch 10-20%, je možné ho vzhledem k výše zmíněným výsledkům považovat za úspěšný.

7 Závěr

7.1 Srovnání jazyků

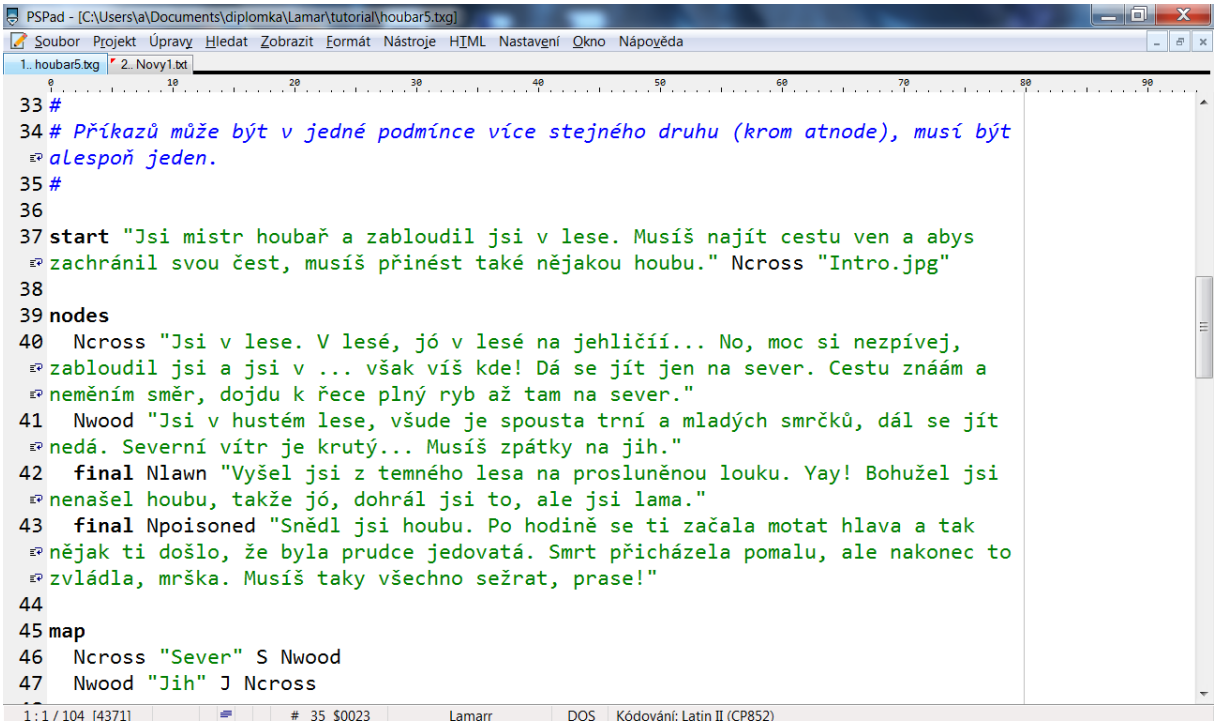
Pro srovnání byly vybrány následující jazyky

7.1.1 HTML+CSS

Striktně nebývá považováno za programování, ale formátování dokumentu. Nedefinuje proměnné, větvení ani cykly, zato umožňuje použití bloků (rozsah platnosti, *scope*) a dědičnost. Je široce rozšířeno (WYSIWYG editory), nevyžaduje žádnou instalaci (poznámkový blok a interpret – webový prohlížeč – jsou automaticky dostupné na všech domácích desktopech). Moderní prohlížeče podporují i ladící konzoli (lze použít i jako ladící IDE) umožňující interaktivně měnit obsah: žáci ihned vidí, jak se změna kódu projeví a odpadají tak fáze psaní – ladění, mezi nimiž žák nemá okamžitou zpětnou vazbu. Lokalizace není implementována. Zobrazovací pravidla nemají vazbu na přímou zkušenost nutnou pro efektivní vývoj na ZŠ (podrobnosti viz kapitola 5.1.1). Z tohoto důvodu může být ideálním prvním jazykem na SŠ (netřeba předchozích zkušeností či vazeb na jiné jazyky), kde již studenti všeobecně disponují schopností formálního uvažování a anglické termíny nevadí.

7.1.2 Lamarr

Byl vyvinut za účelem odstranění překážky HTML. Žáci programují průchod světem, pro který nemají IDE a musí si vystačit se svými představami. Do editorů lze však přidat zvýrazňování syntaxe, v příloze [e1d] je integrace s prostředím PSPad (zvýrazňovač syntaxe, interpret, výchozí soubory). Jazyk umožňuje (ale pro tvorbu her nevyžaduje) i použití proměnných, větvení a procedur. Byl úspěšně otestován na žácích ZŠ, zatímco syntaxe pro práci s předměty je výzvou i pro studenty SŠ. Jazyk se tak hodí pro otestování schopností konkrétní třídy, je vhodný jako úvod do programování v textovém editoru bez použití IDE.



Obrázek 11: Integrace Lamarr do PSPad IDE, zvýrazňovač syntaxe

Jazyk (stejně jako jiný textově orientovaný pro výuku vytvořený jazyk) lze integrovat do OS Windows skriptem popsaným v příloze 8.3. Byl otestován na menším vzorku dětí od 12. let, ale s jednoznačně kladnou odezvou.

7.1.3 Scratch

Z pohledu výuky se tento jazyk blíží dokonalosti ve všech sledovaných ohledech. Je multiparadigmatický (OOP i iterativní paradigma), grafické IDE včetně kolekcí spritů pomáhá zapadajícími tvary rozlišit správnou syntaxi (narozdíl od u nás na školách rozšířeného Baltíka), děti velmi baví, účinně využívá paradigmatu Creative Computing.

Je zdarma, lokalizovaný, má rozsáhlou komunitní podporu a navíc podporuje GPIO Arduina (S4A), tedy je možné použít ho i při výuce automatizace. Autoři na fóru Lifelong Kindergarten Group[13] uvádí použitelnost jazyka Scratch od 8 let, v rámci práce byl úspěšně otestován na 12letých dětech.

[13] https://scratch.mit.edu/parents

Následující jazyky (Javascript, PHP, C) jsou profesionálně používané, není u nich lokalizace, IDE jsou založena na psaní textového kódu, jsou tedy z výšezmíněných důvodů vhodné od SŠ výše.

7.1.4 PHP

Nepodporuje plně objektové paradigma. Program je při každém generování HTML kódu spuštěn, provede posloupnost akcí a skončí, což je iterativní paradigma. Protože podporuje objekty a třídy, je vhodnější ho nazývat *Object Capable* než *Object Oriented*[14].

Díky slabému typování a vyšší úrovni (viz níže) je na školách (např. na SPŠ-VOŠ Písek před změnou ŠVP) používán pro vysvětlení základů objektově orientovaného programování. Vzhledem k výše zmíněným vlastnostem je však vhodné takto vysvětlit pouze syntaxi, nikoliv paradigma OOP.

Jazyky Javascript a PHP vyžadují vyšší míru abstraktního myšlení a hlubší znalosti, např. událost (Javascript) či preprocesor nebo server (PHP).

7.1.5 C

Je silně typovaný, umožňuje práci s pamětí na nízké úrovni, obsahuje STD knihovnu se standardním řešením mnoha situací (knihovna algorithm a další), které plně neovládá ani většina profesionálních programátorů[15]. Jeho možnosti jsou nejširší (vč. psaní jiných jazyků, např. PHP, či operačních systémů), jeho syntaxe je ve zkoumaných jazycích zdaleka nejobtížnější.

Jazyk C vyžaduje technické znalosti, především správu paměti, sestavovací proces (toolchain) a hardware, příp. OS, pro který je program sestavován. Uvedené znalosti musí být na procedurální, nikoliv pouze konceptuální úrovni, tj. nestačí např. znát parametry paměti, je nutno porozumět chybovým hlášením o paměti. Jazyk tedy není

---

[14] Vzácné socketové aplikace jsou většinou psány jako jednoduché smyčky.

[15] Školení C#, 2011

únosné učit jako první. Zmíněné body vyžadují vyšší hodinovou dotaci, je vhodný spíše pro technické SŠ.

7.2 Srovnávací indikátory jazyků

Při srovnání jazyků byly brány v úvahu následující indikátory.

7.2.1 lokalizace

Je možnost psát české příkazy, pokud jazyk využívá IDE, pak také jeho české ovládání, také česká dokumentace k jazyku. Pro žáky ZŠ při prvním vyučovaném jazyku je lokalizace žádoucí, na SŠ pak méně.

- **Scratch a Lamarr** jsou plně lokalizovány
- **HTML+CSS, Javascript a PHP** nelze snadno lokalizovat
- **C** lze lokalizovat pomocí maker

7.2.2 typování

Určuje způsob práce s proměnnými

- *žádné* jazyk proměnné neobsahuje nebo je možné psát základní programy bez použití proměnných
- *fixní* jazyk pracuje pouze s jedním typem proměnných (číslo)
- *slabé* jazyk automaticky převádí typy proměnných
- *silné* typy a jejich převod jsou určeny syntakticky

Z hlediska kognitivních procesů je i silné typování zvládnutelné už na úrovni prvního stupně: znalost čísel a jejich řazení je před nástupem do první třídy zhruba na stejné úrovni jako předčtenářské dovednosti [3, str. 259], pojmy číslo a řetězec jsou známé už dětem na prvním stupni (jedna a jedna je jedenáct, jedna plus jedna jsou dvě). Mladší děti však neudrží pozornost příliš dlouho a nedokáží se koncentrovat na více věcí současně, proto je vhodné jenprve zařadit jazyky s žádným či fixním typováním, po nich pak slabé a pak teprve silné, aby se děti mohly soustředit na program díky tomu, že předchozí stupeň typování už mají zažitý a nemusí mu věnovat pozornost.

- **Scratch** má dva základní typy: číslo a text. Nenabízí přetypování, typ se určí kontextově podle použité operace (porovnání a změna o konstantu implikují číslo).
- **Lamarr** v základních programech nepoužívá proměnné, pouze odkazy na konstanty bez typu. V pokročilých programech operuje pouze s celými čísly (texty jsou vždy konstantní).
- **HTML** neobsahuje proměnné, pouze atributy, což jsou formálně konstanty textového typu; jediná operace s nimi je porovnávání obsahu.
- **Javascript a PHP** obsahují více typů včetně objektů i operátory explicitního přetypování. Ztráta obsahu nezpůsobí chybu programu (nejvýše varování u PHP), k zastavení programu může dojít pouze při volání členu neexistujícího (NULL) objektu.
- **C** je silně typovaný jazyk, při přetypování je nutno znát binární podobu typů v paměti.

7.2.3 názornost

Jako názorný se zde uvažuje jazyk, kde lze po zapsání kódu jednoduchou akcí zobrazit jeho výsledek, který by měl být obecně dostatečně atraktivní.

- **Scratch** ihned po umístění dílku do pole kódu zobrazí jeho výsledek
- **Lamarr** po zapsání kódu stačí předložit jako kód interpretu, výsledkem je textová hra
- **HTML+CSS** po zapsání kódu lze zobrazit v prohlížeči, výsledkem je webová stránka
- **Javascript** není názorný: pro výstup je třeba HTML kód nebo konzole, která nepodává jiný výsledek než textový výpis
- **PHP** je názorný pokud se kód edituje přímo na serveru, ladící hlášení jsou občas nepřehledná (např. při chybě v cyklu)
- **C** není názorný: program je nutno kompilovat, výstupem bývá ve studentských příkladech převážně text

7.2.4 IDE

Integrated development environment udává, je-li použití jazyka vázáno na nějaké vývojové prostředí, bodována byla snadnost zápisu kódu.

- **Scratch** má povinný vstup i výstup pomocí IDE, výsledek lze také automaticky publikovat na webu.
- **Lamarr** je možné integrovat s editorem PS Pad, obejde se však i bez něj.
- **HTML+CSS, Javascript a PHP** potřebují pro názorný výstup prostředí prohlížeče, vstup je možno zadávat libovolným editorem, podpora zvýrazňování syntaxe je velmi rozšířená (PSPad, Sublime, EditPlus, Notepad++ a další).
- **C** má většinu IDE na úroveň studentů příliš rozsáhlé a spíše matoucí. Je možné ho integrovat do editorů umožňující nastavení externích kompilátorů (např výše zmíněné), lze si ale vystačit s obyčejným editorem a dávkovým souborem či nástrojem *make*.

7.2.5 úroveň

Úrovní se u programovacích jazyků myslí jejich vzdálenost od hardware. Čím nižší úroveň jazyka, tím větší požadavky na technické znalosti, čím vyšší úroveň, tím méně je výstup závislý na hardware z pohledu kódu, tj. méně (případně vůbec žádné) příkazy ovládající hardware.

- **Scratch** obsahuje příkazy přímo související s výstupem jako "plachti na pozici".
- **Lamarr** obsahuje intuitivní konstanty, které však musí mít předepsaný formát.
- **HTML** obsahuje značky s pravidly jejich zápisu, **CSS** (obzvláště CSS3) obsahují poměrně komplikované selektory, v krátkém kódu si však lze vystačit s jednoduchým zápisem.
- **Javascript** je obvykle událostmi řízený, **PHP** je obvykle generátor HTML, což je o úroveň abstrakce blíže hardware než předchozí jazyky.

- **C** vyžaduje dobrou znalost práce s pamětí, často lze napsat program, který většinou vrací správný výsledek, ale obsahuje velké množství pro začátečníka těžko odhalitelných chyb, na rozdíl od předchozích jazyků nemá automatickou správu paměti.

	CZ	typování	názornost	IDE	úroveň	typ školy
Scratch	ano	slabé	ano	obojí	velmi vysoká	ZŠ
Lamarr	ano	žádné, fixní	ano	vstup	vysoká	ZŠ
HTML+CSS	ne	žádné	ano	výstup	vysoká	ZŠ/SŠ
Javascript	ne	slabé	ne	výstup	střední	SŠ
PHP	ne	slabé	částečně	výstup	střední	SŠ
C	ano	silné	ne	ne	nízká	SŠ/VŠ

- Výsledky testování HTML+CSS nebyly přesvědčivé ani u studentů 2. ročníku SPŠ-VOŠ Písek, na druhou stranu některé základní školy nabízejí kroužky tvorby webových stránek, tento jazyk byl vyhodnocen jako hraniční mezi ZŠ a SŠ.
- Jazyk C obsahuje mnoho úskalí, na SPŠ-VOŠ Písek je vyučován, jeho procedurálné znalost je však u studentů nízká. Proto je uveden také jako hraniční jazyk.

7.3 Aktualizace ŠVP IT na SPŠ-VOŠ Písek

Při tvorbě ŠVP je nutno zohlednit (závazný) rámec RVP, profilaci školy a autoevaluaci (Národní ústav pro vzdělávání to podrobně specifikuje ve svém doporučujícím manuálu[16]).

Náhled provedených změn z pohledu učiva jedním slovem shrnuje následující tabulka

[16] www.nuv.cz/file/188_1_1

ŠVP IT v SPŠ-VOŠ Písek	do r. 2015 (+dobíhání)	od r. 2015
1. ročník	papír	web-klient (HTML, CSS, JS)
2. ročník	C	web-server, db (PHP, SQL)
3. ročník	PHP	C (pouze specializace)
4. ročník	ST	C# (pouze specializace)

Škola nabízí dva obory

- informační technologie (I)
- elektrotechnika (E)

Do r. 2014 se škola v oboru (I) potýkala s častými změnami, vycházelo se kompromisně především z RVP a kvalifikace učitelů na úkor návaznosti a kognitivních schopností žáků. Hlavní nedostatky původního ŠVP IT zdůvodněné výše jsou především

- nenázornost výkladu algoritmizace (papír)
- zařazení C již ve druhém ročníku
- zařazení ST (IDE MOSAIC) na úkor OOP mající aplikační oblast především v oboru E

Na základě slabých školních výsledků studentů byla svolána komise, která provedla změny respektující výše popsaná fakta.

- v prvních dvou ročnících je splněna část PVA definovaná RVP IT
- od třetího ročníku mají studenti na výběr ze dvou specializací
    - specializace s důrazem na programování a počítačové sítě (vhodná pro analyticky orientované typy)
    - specializace s důrazem na management v IT a informační systémy (vhodná pro prakticky, sociálně orientované typy)

8 Přílohy

8.1 Experimenty

Následující experimenty jsou ve formě jednoduchých úloh a her. Pro účely tvorby závěrů této práce byly prováděny na zkoumaných skupinách trénováním po dobu 2x 3 měsíce (longitudální výzkum) na dvou základních a dvou středních školách (komparace) a porovnání úspěšnosti se studijními výsledky.

8.1.1 [t1] Konceptuální vs. procedurální paměť - dospělí

- 2 čety (celkem 60 vojáků) přenese 5 tun nákladu za 3 hodiny a 20 minut. Za jak dlouho přenese stejný náklad 1 rota (5 čet)? [přímá a nepřímá úměra]
- Po D1 vyjíždí z Prahy do Brna jistý lobbista rychlostí 180 km/h, z Brna do Prahy mu vyjíždí vstříc dálniční policie předpisovou rychlostí 130 km/h. Za předpokladu, že nedojde k havárii, že policie není podplatitelná a že řidič nebude ujíždět: u kterého města budou policisté pana Janouška pokutovat? [lineární rovnice]
- Navrhněte postup a sestavte vzorec výpočtu výšky objektů, na které se nedá vylézt. Máte k dispozici metrovou tyč a úhloměr. [goniometrické funkce]
- Jak daleko dostřelí kulovnice CZ 557? Úsťová rychlost je 800m/s. Střílíme vodorovně ze stoje z výšky 1.5m. [nelineární rovnice, dosazení]
- Navrhněte hrnec (tvar, materiál, struktura), na kterém se dá dobře vařit jídlo a který co nejdéle udrží obsah teplý. Hrnec by měl mít objem tři litry. [úvaha]

8.1.2 [t2] Varianta zkoušky čtení

Přečtěte si následující text a odpovězte na otázky pod ním uvedené. Hodnocena bude rychlost splnění úkolu a správnost odpovědí.

[výňatek z Osmé cesty Ijona Tichého[17]]

17 LEM, S.: *Hvězdné deníky 1*. Baronet, a.s., 1999. ISBN 80-7214-230-5

Zařízení, které podle dohody ratifikované tímto Váženým shromážděním dodalo na základě zevrubných směrnic téže dohody Altairské společenství Šestinásobnému sdružení Fomalhaut, vykazuje vlastnosti – jak se konstatuje v protokole zvláštní podkomise OSP – které nemohou být jen následkem malých odchylek od technologického návodu přijatého oběma stranami. Altairské společenství správně konstatovalo, že jím vyrobené eliminátory záření a planetoreduktory měly sice být schopné reprodukce, zaručující vznik strojového potomstva, což obsahovala platební dohoda obou stran, ale tato potence se měla projevit – podle inženýrské etiky platné v celé federaci – singulárním pučením, a ne obdařením zmíněného zařízení programem s opačnými znaky, k čemuž bohužel došlo. Taková dvojakost programu vedla v oblasti hlavních energetických systémů Fomalhautu ke vzniku vilných antagonismů, jejichž následkem byly morálně pohoršující výjevy, které způsobily žalující straně vážné materiální ztráty. Místo aby se dodané agregáty zabývaly prací, k níž byly zkonstruovány, věnovaly část pracovních směn volbě vhodného partnera, přičemž jejich ustavičné pobíhání se zástrčkami, jehož cílem byl rekreační akt, vedlo k porušení Panundského statutu a ke strojografické explozi, přičemž za oba tyto politováníhodné jevy nese vinu žalovaná strana. Proto se pohledávka Altairu považuje tímto za anulovanou.

Kdo koho žaluje?

Co je předmětem stížnosti?

Co měla obžalovaná strana udělat, aby nebyla bývala žalována?

Vysvětlete, co se v textu myslí *dvojakostí programu*.

Vysvětlete, co se v textu myslí *dodanými agregáty*.

Test byl modifikován ze standardní Zkoušky čtení [5, s.183], přičemž bylo nutno vzít v úvahu rozdíly kódu oproti běžnému textu:

- Kód obsahuje struktury a pojmy, které se v řeči nevyskytují
- Kód se nečte shora dolů, ale "zvenku dovnitř", tj. od hlavního programu k čím dál konkrétnějším podprogramům

Hodnocení: rychlost čtení, počet chyb, stupeň porozumění.

8.2 Elektronické přílohy

Elektronické přílohy jsou na přiloženém USB flash disku s následujícími odkazy do složek v textu práce:

[e1a] /Lamarr

[e1b] /Lamarr/tutorial

[e1c] /Lamarr/Kult tisíce

[e1d] /Lamarr/PSPad

[e2a] /HTML/01 až /HTML/08

[e2b] /HTML/ŠVP

[e3] /C

[e4a] /Scratch

[e4b] /Scratch/zaci

[e4c] /Scratch/zaci2

[e4d] /Scratch/DoD

[e5a] /kompetence

[e5b] /kompetence/cteni

[e5c] /kompetence/matematika

[e6] /C#

[e7] /pisemky

[e8a] /RVP/RVP-ZV.pdf

[e8b] /RVP/RVP-1820M01.pdf

8.3 Integrace jazyka Lamarr do OS Windows

Integraci lze provést spuštěním následujícího dávkového souboru

```
@echo off
assoc .txg=lamarr_txg
ftype lamarr_txg="%CD%\lamarr.exe" "%%1"
cls
echo Install complete. You can run txg files now!
echo.
pause
```

Poté lze bez IDE jazyk používat pomocí přetažení souboru s kódem na zástupce interpreta na ploše.

K dispozici též v elektronické příloze [1]

8.4 Literatura

[1a] *Rámcový vzdělávací program pro základní vzdělávání.* MŠMT, Praha, 2013: Dostupné online: <http://www.nuv.cz/file/318>

Přiloženo jako elektronická příloha [e8a].

[1b] *RVP-18-20-M01 Informační technologie* MŠMT, Praha, 2008. Dostupné online [cit 2015]:<http://zpd.nuov.cz/RVP/ML/RVP 1820M01 Informacni technologie.pdf>

Přiloženo jako elektronická příloha [e8b].

[2] VÁGNEROVÁ, M.: *Vývojová psychologie, Dětství a dospívání*. Karolinum, 2014. ISBN 978-80-246-2153-1.

[3] FISCHER, R.: *Učíme děti myslet a učit se, praktický průvodce strategiemi vyučování*. Portál, 1997. ISBN 80-7478-720-7.

[4] RILEY, GREENO, HELLER: *Development of Children's Problem Solving Ability in Arithmetic*. Learning R&D Center, University of Pittsburgh, 1984. ISBN 0-12-284780-6

[5] SVOBODA, KREJČÍŘOVÁ, VÁGNEROVÁ: *Psychodiagnostika dětí a dospívajících*. Portál, 2001. ISBN 80-7178-545-8

[6] BARNES, D., KÖLLING, M.: *Object First with Java*. Prentice Hall, Fifth edition, 2012. ISBN 978-013-249266-9

[7] TUROŇ, J.: *Projektová výuka programování na SPŠ*, Ostravská univerzita v Ostravě, 2011.

[8] ČSÚ: *IT odborníci podle vzdělání*, 2010. Dostupné online [cit. 2011]: <http://www.czso.cz/csu/redakce.nsf/i/3_it_odbornici_podle_vzdelani>

[9] KRYNICKÝ, M.: *Stav školství v Čechách,* osobní blog, 2008. Dostupné online: <http://www.realisticky.cz/clanky/prilohy/skolstvi/Stav_skolstvi_v_cechach_2008.pdf>

[10] KLEMENT, M. et. al.: *Metody realizace a hodnocení výuky základů programování*, Olomouc 2012. ISBN 978-80-87658-01-7

[11] LAPLANTE, P., NEILL, C.: *Antipatterns: Identification, Refactoring and Management*, Auerbach Publications 2005. ISBN 0-8493-2994-9