Univerzita Karlova v Praze

Pedagogická fakulta

Katedra informačních technologií a technické výchovy

Problematika algoritmizace a programování v rámci předmětu Informatika na základních školách

Autor: Bc. Karel Albrecht

Vedoucí diplomové práce: Mgr. Tomáš Jeřábek

Studijní program: Učitelství pro střední školy

Praha 2014

UNIVERZITA KARLOVA V PRAZE
PEDAGOGICKÁ FAKULTA
Katedra informačních technologií a technické výchovy

ZADÁNÍ DIPLOMOVÉHO ÚKOLU

akademický rok 2012/2013

Jméno a příjmení studenta: Karel Albrecht

Studijní program: N7504 Učitelství pro střední školy

Studijní obor: Učitelství VVP pro ZŠ a SŠ – informační a komunikační technologie

Název tématu práce v českém jazyce:

Problematika algoritmizace a programování v rámci předmětu Informatika na základních školách

Název tématu práce v anglickém jazyce:

Pokyny pro vypracování:

- analyzovat dostupnou literaturu a další informační zdroje z oblasti výuky programování na ZŠ
- analyzovat české a zahraniční vzdělávací programy z hlediska začlenění algoritmizace a programování do výuky informatických předmětů na 2. stupni vzdělávání a vzájemně je porovnat
- zmapovat možnosti podpory výuky programování na ZŠ z hlediska technologického a didaktického, zejména na úrovni technologií a specifických softwarů
- navrhnout vhodnou koncepci výuky programování s využitím vybraného programovacího či skriptovacího jazyka pro jedno školní pololetí pro vybraný ročník 2. stupně ZŠ
- provést výzkumné šetření navržené koncepce výuky v podobě akčního výzkumu v rámci jednoho školního pololetí na vybrané ZŠ
- zhodnotit přínos a nedostatky navržené koncepce a vyvodit doporučení pro následné zlepšení

Vedoucí diplomové práce: Mgr. Tomáš Jeřábek

Předpokládaný rozsah diplomové práce[1]: 60 normostran

Datum zadání práce: 7.11. 2013

Předběžný termín odevzdání práce: říjen 2014

V Praze dne:

.....
doc. PhDr. Vladimír Rambousek, CSc.
vedoucí katedry

[1] Minimální rozsah diplomové práce je standardně 60 normostran (108 000 znaků vč. mezer) vlastního textu.

Prohlašuji, že jsem diplomovou práci na téma Problematika algoritmizace a programování v rámci předmětu Informatika na základních školách vypracoval pod vedením vedoucího diplomové práce samostatně za použití v práci uvedených pramenů a literatury. Dále prohlašuji, že tato diplomová práce nebyla využita k získání jiného nebo stejného titulu.

Datum: 18. 6. 2014

.....

podpis

Rád bych touto cestou vyjádřil poděkování
Mgr. Tomáši Jeřábkovi za jeho cenné rady a
trpělivost při vedení mé diplomové práce.

.....

podpis

NÁZEV:
Problematika algoritmizace a programování v rámci předmětu Informatika na základních školách

AUTOR:
Bc. Karel Albrecht

KATEDRA:
Katedra informačních technologií a technické výchovy

VEDOUCÍ PRÁCE:
Mgr. Tomáš Jeřábek

ABSTRAKT:
Práce se zabývá problematikou algoritmizace a programování na základních školách. Hlavním cílem celé práce je návrh nové koncepce programování, která by měla přinést lepší výsledky v oblasti programování a efektivnější způsob výuky. V teoretické části jsou uvedeny nejdůležitější pojmy z oblasti algoritmizace a programování, analýza českých a zahraničních vzdělávacích systémů, zmapování možností podpory výuky z technologického a didaktického hlediska. Z teoretických východisek je navržena nová koncepce programování. Ta je ověřena zhruba půl ročním akčním výzkumem při výuce programování na 2. stupni ZŠ. V závěru práce jsou uvedeny zkušenosti, klady a zápory nové koncepce výuky programování.

KLÍČOVÁ SLOVA:
programování, algoritmus, JavaScript

TITLE:
The issue of algorithm development and programming within the scope of computer science at primary schools

AUTHOR:
Bc. Karel Albrecht

DEPARTMENT:
Department of information technology and education

SUPERVISOR:
Mgr. Tomáš Jeřábek

ABSTRACT:
The thesis is concerned with problems of algorithm development and programming on primary schools. The main target of the whole thesis is the project of new programming conception, which should bring better results on the field of programming and more effective way of teaching. In theoretic part there are introduced the most important terms from the field of algorithm development and programming, analysis of Czech and foreign education systems, mapping of possibilities of support of teaching from technological and didactical viewpoint. The new programming conception is suggested from theoretical bases. That conception is verified by approximately half-year action research at teaching of programming on 2nd level of primary school. In conclusion of the thesis there are introduced experiences, plusses and minuses of new conception of programming teaching.

KEYWORDS:
programming, algorithm, JavaScript

Obsah

1. Úvod

Se stále vyšším podílem IT techniky a jejími inovacemi ve všech oborech roste i potřeba vzdělávání v této oblasti. Podle některých prognóz bude v roce 2016 nedostatek kvalifikovaných pracovníků v oblasti IT.[1] Zároveň s tímto trendem vyššího podílu IT techniky ve všech oblastech roste tlak na zvládání algoritmického myšlení, které umožňuje efektivnější práci s touto technikou.

Práce se zabývá problematikou výuky programování a algoritmizace na ZŠ, která částečně přispívá právě k rozvoji tohoto myšlení. Snaží se v jednoduchosti popsat základní pojmy spojené s problematikou programování, jako je algoritmizace, algoritmus, program, programování. Následně rozděluje programovací jazyky z hlediska výuky do dvou skupin a uvádí různé příklady těchto jazyků.

Zabývá se výukou programování na 2. stupni ZŠ z hlediska právního rámce a z pohledu konkrétních možností výuky programování v České republice. Dále uvádí možnosti programování a algoritmizace ve vybraných státech světa, získané především z kurikulárních dokumentů těchto států.

Práce dále analyzuje různé koncepce z hlediska organizačních forem, metod a prostředků, na jejichž základě stanovuje dvě základní používané koncepce výuky programování a algoritmizace. Jednotlivé koncepce následně analyzuje z hlediska jejich pozitiv a negativ při výuce programování a algoritmizace.

Z analýzy výše uvedených koncepcí a přístupů k výuce programování a algoritmizace se snaží navrhnout novou optimální koncepci, která by byla vhodná pro výuku programování a algoritmizace na ZŠ.

Navržená koncepce je v průběhu půl roku vyzkoušena v předmětu informatika na běžné základní škole a kvalitativním výzkumem je zhodnocena. Výzkum slouží k zodpovězení základních otázek, které jsou na jeho počátku zformulovány a k potvrzení či vyloučení hypotézy, která vyplývá z otázek.

V závěru práce jsou zhodnoceny jednotlivé kroky vedoucí k návrhu nové koncepce a zároveň jsou shrnuty výhody a nevýhody nově navržené koncepce.

[1] *Do roku 2016 na západních trzích nastane nedostatek IT odborníků* [online]. Businessinfo.cz [cit. 2013-12-30] Dostupný z WWW: <https://www.businessinfo.cz/cs/clanky/do-roku-2016-na-zapadnich-trzich-nastane-nedostatek-it-odborniku-31954.html>.

2. Cíle a úkoly práce

Mezi hlavní cíle této práce patří návrh nové koncepce výuky programování či algoritmizace, pomocí níž by měla výuka této oblasti být efektivnější a měla by dosahovat lepších výsledků. Způsobem řešení je navrhnout vlastní koncepci, která vychází z analýzy a inovace známých metod, forem a prostředků se současným využitím elektronické podpory. K naplnění hlavního cíle bude sloužit výzkum v podobě akčního výzkumu trvajícího přibližně jedno pololetí na vybrané ZŠ, kde bude porovnávána nově navržená koncepce s již dříve používanou. Pro realizaci koncepce a následného výzkumného šetření bude vybrán na základě předchozí analýzy nejvhodnější programovací či skriptovací jazyk, který se do prostředí dané školy hodí.

Mezi dílčí cíle sloužící k naplnění základního cíle patří zmapování možností podpory výuky programování na ZŠ z technologického a didaktického hlediska, což přinese větší možnost výběru použitých technologií a metod při návrhu celé koncepce. Z pohledu technologického a didaktického je dalším cílem analyzovat české a zahraniční vzdělávací programy právě z hlediska začlenění algoritmizace a programování v jejich obsahu. Při návrhu nové koncepce brát v úvahu zmíněné možnosti algoritmizace a programování v jednotlivých státech. Posledním cílem je zhodnotit přínosy a nedostatky navržené koncepce a vyvodit doporučení pro následné zlepšení.

K naplnění části úkolů bude využit výzkum teoretický a to především analýza. Dále pak výzkum empirický například pro zpracování pozorování, rozhovorů, didaktických testů a dotazníků.

V práci budou nejprve vymezeny základních pojmy související s programováním a algoritmizací. Dále proběhne analýza českých a zahraničních vzdělávacích systémů, analýza využitelných metod, forem a prostředků při výuce programování a algoritmizace. Po analýze dostupných koncepcí výuky programování a algoritmizace bude proveden návrh vlastní koncepce na základě předchozích analýz. Navržená koncepce bude poté ověřena v praxi a budou z ní vyvozeny závěry.

3. Teoretická východiska

V části teoretických východisek jsou popsány základní pojmy a základní klasifikace programovacích jazyků, které lze využít při výuce této problematiky na druhém stupni ZŠ. Další část této kapitoly popisuje legislativní oporu výuky programování v České republice a dalších vybraných státech světa s příklady jejich možného využití v praxi. Poslední část je věnována koncepcím výuky programování a algoritmizace, které se objevují ve výuce v ČR a ve světě.

3.1. Vymezení základních pojmů

Problematika programování a algoritmizace s sebou přináší několik pojmů a základní rozdělení programovacích jazyků, které je vhodné uvést. Proto tato kapitola bude věnována právě této problematice. Mezi základní pojmy, které jsou často dále uváděny, patří:

- **Algoritmizace**
- **Algoritmus**
- **Programování**
- **Program**

Dále se práce často odkazuje na různé programovací jazyky a jejich základní dělení, mezi které patří:

- **Propedeutické programovací jazyky**
    - **Textově orientované**
    - **Stavebnicové**
    - **Kartičkové**
    - **Ikonické**
- **Klasické programovací jazyky**
    - **Nižší**
    - **Vyšší**
        - **Procedurální**
        - **Neprocedurální**

3.1.1. Algoritmizace, algoritmus

Algoritmizace je postup, pomocí kterého lze vyřešit určitý problém. V případě práce na počítači se jedná o vytváření postupu řešení problému právě na něm. Při algoritmizaci

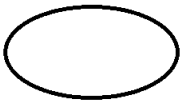
vytvářím postupy, při kterých využíváme svůj intelekt a zkušenosti. Prvními pokusy se většinou k cíli nedostaneme, ale postupnou úpravou vytvářím stále lepší algoritmy.[2]
Algoritmizace by se v čase měnit neměla, to však neplatí o programovacích jazycích, které zastarávají a jsou nahrazovány jinými.
Pojem algoritmizace je mnohem širší, nežli bylo uvedeno v předchozích odstavcích. Jeho využití se nevztahuje pouze na výpočetní techniku, ale algoritmizaci jako takovou lze využít ve všech předmět vyučovaných na ZŠ a odhlédneme-li od školy, tak i v mnoha oblastech běžného života.
Výsledkem algoritmizace úloh může být algoritmus. Je to vlastně předpis, který se skládá z jednotlivých kroků zabezpečujících na základě vstupních dat data výstupní. Každý algoritmus by měl mít základní vlastnosti:

- **Determinovanost:** Přesně musí být stanoveno, jak jdou jednotlivé kroky za sebou a musí být patrno, jaký bude další krok.
- **Hromadnost:** Algoritmus většinou neřeší jen konkrétní situaci, ale lze ho obecně použít v podobných vhodných situacích.
- **Opakovatelnost:** Stejné vstupní údaje musí poskytovat stejné výstupní údaje.
- **Rezultativnost:** Měl by vést k výsledkům ať už správným či špatným.[3]

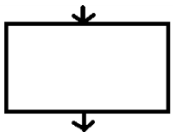
Ke znázornění algoritmu lze využít vývojový diagram, který obsahuje několik symbolů s konkrétním významem. Pomocí těchto symbolů lze složit celý diagram, řešící konkrétní problematiku. Mezi základní používané symboly patří[4]:

[2] *Problematika algoritmizace.* Mendelova univerzita [cit. 2013-12-30] Dostupný z WWW: <https://akela.mendelu.cz/~mot/vyuka/skralg01.pdf>.
[3] DOHNAL, Pavel. *Programování na základních školách.* Diplomová práce. [cit. 2013-12-30] Dostupný z WWW: <http://is.muni.cz/th/72886/pedf_m/Diplomova_prace_Pavel_Dohnal.pdf>.
[4] TAUFEN, KOTYK, HRUBINA. *Algoritmy a algoritmizace – vývojové diagramy.* Pardubice: Univerzita Pardubice fakulta informatiky a elektrotechniky. ISBN 978-80-7395-182-5.

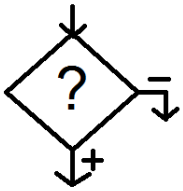
- **Začátek a konec algoritmu**



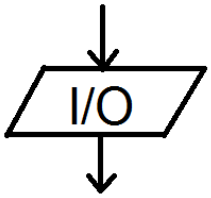
- **Symbol příkazů - zpracování**



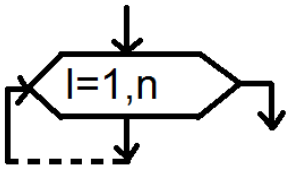
- **Symbol rozhodování**



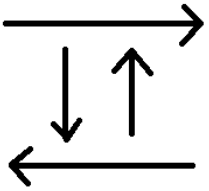
- **Symbol vstupu nebo výstupu**



- **Symbol pro cyklus se známým počtem opakování**



- **Řídící tok (šipky)**



Obrázek 1: Nejčastěji používané symboly vývojových diagramů

V užším pojetí je pojem algoritmizace často spojován s tvorbou algoritmů, které jsou následně počítačově zpracovávány. Algoritmizaci počítačově zpracovávaného problému lze rozdělit do pěti kroků. Prvním krokem je formulace problému, kde je potřeba formulovat požadavky, vstupní údaje a výstupní údaje včetně jejich přesnosti. Druhým krokem je analýza problému, kdy se zjišťuje, jeli vůbec řešitelný a zdali jsou vstupní hodnoty dostačující. Na základě druhého kroku se autor snaží nalézt co možná nejjednodušší řešení problému. Třetím krokem je samotné vytvoření algoritmu, ve kterém, jak bylo popsáno výše, dojde k sestavení sledu jednotlivých instrukcí vedoucích k správnému řešení úlohy. Čtvrtým krokem algoritmizace je sestavení programu, což je v tomto pojetí přepis sestaveného algoritmu do některého z programovacích jazyků, který následně zprostředkuje vykonání kroků na počítači či jiném elektronickém zařízení.

Posledním krokem algoritmizace je odladění programu, které přináší odstranění syntaktických a logických chyb. Syntaktické chyby jsou většinou programem rozpoznány (chyby v zápise) naproti tomu logické chyby rozpoznány být nemusí, což vede ke špatné činnosti programu. Tyto chyby lze nalézt ladícími programy, pomocí kterých lze sledovat velké množství hodnot souvisejících s během programu.[5]

3.1.2. Programování, program

Programování je činnost člověka vedoucí k tvorbě programu. Programování je složeno z několika kroků, které na sebe navazují. Prvním krokem je správné pochopení úlohy, úvaha nad možnými situacemi a rozhodnutí, jaká data budou vstupní a jaká výstupní. Následuje sestavení samotného algoritmu. V případě složitějšího algoritmu, je možné rozdělit činnosti do úseků, které budou řešeny samostatně. V dalším kroku následuje přepis vytvořeného algoritmu do zvoleného programovacího jazyka. Posledním krokem by měla být úvaha nad hotovou prací a její možné vylepšení. Výsledkem celého programování se stává program zapsaný v konkrétním jazyce.[6]

Program je již hotový přepis algoritmu do konkrétního programovacího jazyka. Program se nevytváří sám, ale je výsledkem práce programátora, který obsah algoritmu přetransformoval do programovacího jazyka. Ke spuštění programu je následně zapotřebí procesor, který tento program zpracuje. Mezi základní znaky programu patří práce se zadanými daty, přesně stanovené operace s těmito daty a z nich vycházející výsledky. Výsledkem lze chápat přetransformovaná data pomocí příkazů programu.[7]

3.1.3. Programovací jazyky

Programovací jazyk je formální jazyk sloužící pro zápis algoritmu přesně definovanou syntaxí. Existuje řada dělení těchto jazyků. Pro potřeby této práce lze rozdělit jazyky do dvou skupin a to na propedeutické a klasické.

[5] BECHYŇOVÁ, *Marta. Stránky k výuce informatiky-Algoritmizace*. [online]. [cit. 2014-05-23] Dostupný z WWW: <http://www.ivt.mzf.cz/algoritmizace-a-programovani/uvod-do-algoritmu/2-algoritmizace/>.
[6] DVORSKÝ, ĎURÁKOVÁ, OCHODKOVÁ. *Základy algoritmizace*. Ostrava: Technická univerzita, 2004.
[7] MARTIŠEK, Dalibor. *Algoritmizace a programování v Delphi*. 1. vyd. Brno: Littera, 2007.

3.1.4. Propedeutické programovací jazyky

Jazyky spadající do této kategorie slouží především k snadnějšímu pochopení problematiky programování a jejímu lehčímu zvládnutí. Z pohledu dětí, které vyžadují rychlé výsledky, se dostavuje také motivační efekt. V některých programech lze po pár minutách práce dosáhnout velmi rychle grafické a zvukové výstupy, jejichž dosažení v klasických jazycích trvá většinou déle. Důležitým aspektem, hlavně u mladších žáků je dostupnost těchto programů v českém jazyce, takže při jejich ovládání nemusí řešit problematiku překladů. Tyto jazyky jsou velmi vhodné jako začátek programování před jazyky klasickými, byť některé z nich nabízí možnosti i klasických jazyků.

Dalším aspektem těchto jazyků je umožnění práce v podstatě každému. Každý, bez ohledu na věk, by měl být s programem schopen pracovat. Je zde pouze spodní věková hranice, která je dána schopností dítěte ovládat nebo číst a orientovat se v daném prostředí. Posledním důležitým aspektem je možnost využití tohoto prostředí i pro ty dobré, kteří jsou schopni vytvořit i složitější algoritmy.

Většina propedeutických jazyků se dá rozdělit do čtyř skupin. První skupina jsou textově orientované programovací jazyky, které se vyznačují tím, že pro jejich ovládání je důležitý zápis kódu, pro který platí syntaktická pravidla. Mezi hlavní zástupce tohoto směru patří jazyk LOGO, Imagine LOGO, Comenius LOGO a jejich další velká variace. Druhou skupinu tvoří stavebnicové programovací jazyky, které se vyznačují jednotlivými dílky, které reprezentují příkazy nebo výrazy. Při jejich sestavování zapadnou do sebe pouze ty, které dávají smysl a mohou být v programu takto řazeny. To neznamená, že nelze udělat chybu, jen je tím předcházeno velkým chybám. Stavebnicově vytvořený program je velmi přehledný a jednoduše se upravuje. Hlavním představitelem stavebnicově orientovaných jazyků je Scratch. Třetí skupinu tvoří kartičkové jazyky, kde se program vytváří z předem připravených částí. Každou část lze libovolně umístit a nastavit množství opakování. Mezi zástupce třetí skupiny patří jazyk Squeak a Etoys, které nejsou dostupné v české verzi. Poslední čtvrtou skupinu tvoří ikonické jazyky. Ikonické jazyky se snaží eliminovat zápis programu převodem na jeho ikonické zobrazení, u kterého lze nastavovat parametry jednotlivých ikon. Program ve výsledku vypadá jako obrázek, který je tvořen jednotlivými

ikonami. Každá ikona přitom může vypadat jinak a tím se otevírá možnost i pro děti, které ještě neumí číst. Rozšířenými zástupci této poslední oblasti je Lego Mindstorms, Baltík.[8]

Mezi propedeutické programovací jazyky, které lze využít, patří již od 80. let programovací jazyk Karel, kterého zvládnou ovládat i menší děti. V současné době ho lze využívat i v on-line verzi.[9] Další možností je program Baltík, který není sice zdarma, ale byl v minulosti ve školách více používaný a i metodika výuky byla propracovanější. V současné době jsou nabízeny 3 verze. SGP Baltík 3 je od 4 let, SGP Gems od 6 let a verzí využitelnou v 8. nebo 9. ročníku je SGP Baltík 4 C#. Poslední uvedená verze je založená na C#, Direct X a .Net podporující prostorové programování pro operační systémy Windows.[10]

Mezi programy, které jsou ve světě využívanější a do češtiny přeložené, patří placený program Imagine LOGO, který je určen právě pro začátky výuky programovaní. Jedná se o objektově orientovaný jazyk, který využívá tlačítka, překrývající se plochy, multimédia, síťové prostředí a mnoho dalších moderních prvků. Jeho velkou výhodou je možnost nejen tvorby samotného programu, ale i tvorby prezentací, které umožňují pohyb s animovanými i neanimovanými prvky, což umožňuje jeho využití i v jiných předmětech.[11] K Imagine LOGO se dá dohledat velké množství didakticky zpracovaných příkladů, které se dají využít při výuce. Zároveň se dají sehnat ucelené návody a knihy, jak s daným programem pracovat.

Mezi programy, které jsou taktéž přeložené do češtiny a v jiných státech využívány patří Scratch. V současné době ho lze využít v on-line verzi bez nutnosti instalace. V nynější ustálené podobě (např. Scratch 2.0 Offline Editor z dubna 2014) byl Scratch uveden v roce 2007, tudíž se jedná o jeden z nejnovějších jazyků. Na jeho vývoji se podílel Mitchel Resnick a jeho skupina na MIT. Cílem bylo vytvořit komunitu, kde děti budou moci sdílet

[8] BÉLIK, Miloš. *Malé programovacie jazyky* [online]. 18. 4. 2011.[cit. 2013-12-31] Dostupný z WWW: <http://sopusik.wordpress.com/2011/04/18/19-male-programovacie-jazyky/>.
[9] MALÝ, Martin. *Jak vést děti k programování* [online]. [cit. 2013-12-30] Dostupný z WWW: <http://www.lupa.cz/clanky/jak-vest-deti-k-programovani/>.
[10] SGP systems. *Výukové programovací nástroje* [online]. [cit. 2013-12-30] Dostupný z WWW: <http://www.sgpsys.com/cz/Products.asp>.
[11] Imagine logo. *Imagine LOGO programování pro děti* [online]. [cit. 2013-12-30] Dostupný z WWW: <http://www.pf.jcu.cz/imagine/index.php>.

své interaktivní příběhy[12]. Pracovat ve Scratch by měly zvládnout děti od 8 let, avšak nebývá problém v tomto prostředí pracovat i s mladšími dětmi.[13]

3.1.5. Klasické programovací jazyky

Na rozdíl od propedeutických programovacích jazyků jsou klasické programovací jazyky většinou používané programátory na profesionální úrovni k tvorbě programů. Nasazení těchto jazyků při výuce programování bývá také velmi časté, ale stejně jako u výše zmíněných jazyků, je důležité vybrat ten správný.

Mezi základní rozdělení klasických programovacích jazyků patří dělení na nižší a vyšší programovací jazyky. Programování v nižších jazycích je přizpůsobeno konkrétnímu druhu procesoru, programování je poměrně složité nicméně efektivní. Pro zápis se používá assembler nebo strojový kód. Z hlediska programování na 2. stupni ZŠ a současným trendům (které jsou spíše zaměřeny na vyšší programovací jazyky) je zbytečné o tomto způsobu programování na této úrovni uvažovat.

Pro žáky na 2. stupni ZŠ jsou vhodnější vyšší programovací jazyky, které se opět z různých pohledů a paradigmat dají rozdělit. Důležitým dělením z pohledu následné výuky je dělení na procedurální a neprocedurální. Procedurální jazyky umožňují velmi dobrou návaznost na připravený algoritmus a jeho přepis do konkrétního jazyka, který je zapsán v přesné posloupnosti. Mezi tyto jazyky zejména patří Pascal (který byl původně vytvořen pro výuku programování), C, Basic, PHP, Python (někdy uváděn v neprocedurálních) a další například objektově orientované C++, Java, JavaScript, C#. K jazykům neprocedurálním, které se většinou ve výuce nepoužívají, patří Prolog, Lisp a další.[14]

Programovací jazyk Delphi / Object Pascal je oblíbeným programovacím jazykem, který byl firmou Borland uveden na trh v roce 1995. Jedná se v uvozovkách o spojení vývojového prostředí Delphi, které umožňuje využití velké řady vizuálních nástrojů pro vytváření všech částí aplikace s objektově orientovaným jazykem Object Pascal. Díky této kombinaci mohou žáci provést poměrně rychle a jednoduše návrh celého programu

[12] RESNICK, Michel. *Biography.* MIT Media Lab [online]. [cit. 2013-12-30] Dostupný z WWW: <http://www.media.mit.edu/people/mres>.
[13] Scratch. Vytvářej příběhy, hry a animace Sdílej s ostatními z celého světa [online]. [cit. 2014-05-22] Dostupný z WWW: < http://scratch.mit.edu/parents/>
[14] ŠALOUN, Petr. *Základy programování.* Ostrava: Vysoká škola báňská. [cit. 2014-01-01] Dostupný z WWW: <http://homel.vsb.cz/~s1a10/educ/ZP/prednasky/13-prog-jaz.pdf>.

(aplikace) a rychle se dostanou k efektnímu výsledku.[15] V dnešní rychle se měnící době, převážně v oblasti používání operačních systému, dochází i k rychlému sledu nových verzí tohoto programovacího jazyka. Velice oblíbenou verzí byla verze Delphi 7, která byla pro školy v základní verzi dostupná zdarma a dodnes je pro základy programování v operačním systému Windows využitelná. V současné době je nejnovějším produktem verze Delhi XE5, která umožňuje vývoj nativních aplikací pro nejpoužívanější platformy (Windows, Mac, Android, iOS).[16]

Java je dalším možným jazykem, byť její čistě objektově orientovaná syntaxe a sémantika by žákům mohla činit problémy. V její prospěch naopak hovoří její veliká rozšířenost a využití v podstatě na všech platformách. Při tvorbě programu lze vytvářet buď programy pro nejpoužívanější platformy nebo aplety s využitím na webových stránkách. Tvorba programu probíhá ve dvou krocích. Prvním krokem je samotný zápis zdrojového kódu, ke kterému postačí jednoduchý textový editor nebo editor zvýrazňující syntaxi. Druhým krokem je překlad do takzvaného bajtkódu, který je následně spustitelný na javovském virtuálním stroji (JVM). Sada nástrojů sloužící k programování v jazyce Java se nazývá Java Development Kit (JDK). V současné době se nachází ve verzi 7 a lze ji získat zdarma.[17]

Další možnosti vychází z webových platforem od značkovacího jazyka HTM, CSS až po JavaScript, který bývá nazýván jazykem, skriptem, JavaScriptovým programem či jinak. Je to dáno jeho charakterem a jeho využitím. Jedná se o objektově orientovaný jazyk, který by samostatně nepracoval, pokud by nebyl umístěný jako součást webové stránky nebo se stránka na něj přímo neodkazovala. Lze konstatovat, že se jedná o jazyk doplňkový. Při jeho implementaci do stránek prohlížeči pouze sdělíme, kde JavaScript začíná a kde končí. Pro jeho použití není potřeba žádný speciální program, není potřeba instalovat žádný programovací jazyk, jakož tomu bylo u předchozích jazyků. Stačí, aby operační systém obsahoval webový prohlížeč, který JavaScript podporuje, což je naprostá většina prohlížečů. Samotný kód jazyka stačí umístit do souboru s webovou stránkou a spustit ve

[15] KADLEC, Václav. *Učíme se programovat v delphi a jazyce Object Pascal*. Vydání 1. Praha: Computer Press, 2001. 288 s. ISBN 80-7226-245-9.
[16] Embarcadero Technologies. *History of Delphi Innovation* [online]. [cit. 2014-01-02] Dostupný z WWW: <http://www.embarcadero.com/landing-pages/delphi7-to-rad-xe5>.
[17] TRONÍČEK, Zdeněk. *Učebnice jazyka Java.* 30. 9. 2011. [cit. 2014-01-02] Dostupný z WWW: <http://java.cz/article/ucebnicejazykajava>.

webovém prohlížeči. K zápisu jazyka stačí jednoduchý textový editor či editor zvýrazňující syntaxi typu HomeSite.[18]

PHP je programovací jazyk, který se ve většině případů používá jako skriptovací jazyk běžící na straně serveru. Slouží k tvorbě automaticky generovaných stránek s proměnlivými informacemi v čase. Z tohoto pohledu pro prvotní použití bez předchozích znalostí webových technologií nebo programování moc vhodný není.[19] S přihlédnutím k již výše zmíněné studii (Kotek, 2013) se jedná o nejrozšířenější doplňkový programovací jazyk na SŠ, tudíž pokud předcházející výuka splňuje zmíněné požadavky, lze ho vyučovat. V jeho prospěch také hovoří cena jeho pořízení, která je zdarma. Aby programování v tomto jazyce mělo smysl a výstupy, je dobré místo instalace na jednotlivé stroje, PHP nainstalovat na webový server, přes který si žáci výsledek své práce prohlíží. Druhou variantou je využití některých free hostingových služeb, které často podporu PHP mají. Při tomto využití nejsou žáci vázáni na techniku ve škole, ale mohou na svých pracích procovat z místa s přístupem k internetu. V současné době lze PHP včetně dalších programům sloužících k instalaci webového serveru podporujícího tuto technologii stáhnout pro operační systémy Windows, Mac OS X, a různá sestavení Linuxu. Nejnovější dostupnou verzí je 5.5.7.[20]

Samozřejmě existuje další velká řada programovacích jazyků, které zde zmíněny nebyly a mohou mít své opodstatnění k nasazení ve výuce.

3.2. Programování a algoritmizace na ZŠ v ČR a vybraných státech světa

V jednotlivých částech světa je problematika programování a algoritmizace vnímána s jinou prioritou a jiným způsobem. Ve středoevropských zemích je tendence výuky programování směřována spíše k výuce algoritmického myšlení, které lze následně využít v běžném životě. Oproti tomu v Anglii je výuka chápána jako příprava pro životní uplatnění právě v oblasti IT, které by v budoucnu mělo mít nedostatek pracovních sil.

---

[18] ŠKULTÉTY, Rastislav. *JavaScripr programujeme internetové aplikace*. Vydání 2. Brno: Computer Press, 2004. 224s. ISBN 80-251-0144-4.

[19] KOSEK, Jiří. *PHP tvorba interaktivních internetových aplikací.* Vydání 1. Praha: Grada, 1998. 492s. ISBN 80-7169-373-1.

[20] *Downloads.* [online]. The PHP Group. [cit. 2014-04-01] Dostupný z WWW: <http://php.net/downloads.php>.

3.2.1. Právní rámec v ČR

Základy programování a algoritmizace na základních školách v české republice vycházejí z rámcového vzdělávacího program (RVP), který se o této problematice zmiňuje ve vzdělávací oblasti Informační a komunikační technologie a dále pak v obecné rovině z pohledu klíčových kompetencí, které by měl žák po absolvování základní školy získat.[21]

Jak z RVP vyplývá, každý žák by měl po absolvování základní školy získat elementární znalosti v ovládání výpočetní techniky a moderních informačních technologií. Měl by se rychle a efektivně orientovat ve světě informací a s nimi umět pracovat a využívat je v praktickém životě. S těmito požadavky souvisí i cílové zaměření vzdělávací oblasti, které si jako jeden z cílů klade „schopnosti formulovat svůj požadavek a využívat při interakci s počítačem algoritmické myšlení“. Z tohoto cíle, který jako jediný v této oblasti zmiňuje algoritmické myšlení, je možno problematiku programování zařadit do školních vzdělávacích programů (ŠVP). Vzhledem k obecné formulaci výše zmíněného cíle „výuky algoritmizace“ v RVP je jen na rozhodnutí školy, zdali problematiku programování do ŠVP zařadí, či ne. Jak z definice pojmu algoritmizace vyplývá, škola má taktéž možnost „algoritmické myšlení“ vyučovat v rámci jiných předmětů, kde se tento způsob myšlení také uplatňuje.

Další z cílů oblasti informační a komunikační technologie nabízející prostor pro nasazení programování a algoritmizace na ZŠ je „pochopení funkce výpočetní techniky jako prostředku simulace a modelování přírodních i sociálních jevů a procesů“. Zde má škola možnost využít programy dodávané na řešení této problematiky nebo ve vhodných situacích využít programovací jazyky umožňující vytvoření simulace například přírodních jevů a procesů.

Dalším cílem výše zmíněné oblasti, ve kterém by se taktéž dala najít možnost využití programovacích jazyků na ZŠ, je „tvořivé využívání softwarových a hardwarových prostředků při prezentaci výsledků své práce“. Při prezentaci žákovské práce, je někdy vhodné využití některého z programovacích jazyků pro lepší prezentaci, nežli využití klasických prezentačních softwarů.

Algoritmické myšlení taktéž umožňuje lepší dosáhnutí dalšího cíle, kterým je „využívání výpočetní techniky, aplikačního i výukového software ke zvýšení efektivnosti své učební

[21] *Rámcový vzdělávací program pro základní vzdělávání.* Praha: MŠMT, 2013. [cit. 2013-12-16] Dostupný z WWW: <http://www.msmt.cz/file/29397/download/>.

činnosti a racionálnější organizaci práce“, kde z jednou z možností aplikačního softwaru může být i programovací software.

Mezi klíčové kompetence z RVP, k jejichž dosahování by taktéž měla přispívat oblast informační a komunikační technologie patří „kompetence k učení“. Problematiku algoritmizace a programování lze spatřovat v tom, že žák má „vyhledávat a třídit informace a na základě jejich pochopení, propojení a systematizace je efektivně využívat v procesu učení, tvůrčích činnostech a praktickém životě“. Pro třídění a systematizaci informací je dobré mít algoritmické myšlení, které lze získat právě výukou algoritmizace a programování. S touto problematikou taktéž souvisí kompetence k řešení problémů, kde při jejím zvládnutí by žák měl „samostatně řešit problémy, volit vhodné způsoby řešení, užívat při řešení problémů logické, matematické a empirické postupy“. Z kompetencí komunikativních, by to byla „formulace a vyjádření své myšlenky a názoru v logickém sledu, vyjadřování se výstižně, souvisle a kultivovaně v písemném i ústním projevu“.

Jak z výše uvedené problematiky vyplývá, není nutností při výuce oblasti informační a komunikační technologie zařazovat přímo do výuky algoritmizaci či programování. V souvislosti s dnešní dobou a s generací nazývající se „Síťová generace“,[22] žijící v prostředí neustálých inovací, je důležité se nad touto problematikou zamyslet a zvážit, jestli právě zařazení této oblasti nepřinese žákům a společnosti větší uplatnění v životě a na pracovním trhu.

3.2.2. Využívané možnosti algoritmizace a programování na 2. stupni ZŠ

Jak bylo již uvedeno, není nutností algoritmizaci či programování na školách vyučovat, ale i tak se najde řada škol, která tuto problematiku do výuky zařadila. Vzhledem k tomu, že školství v ČR nemá systematickou podporu programování, je v této kapitole naznačeno, jakým směrem a jaký způsob algoritmizace či programování mohou školy případně zařadit do ŠVP:

- **Výuka robotizace**
- **Výuka programování**
    - **Výuka propedeutického jazyka**
    - **Výuka klasického jazyka**

[22] BRDIČKA, Bořivoj. *Síťová generace podle Tapscotta* [online]. Metodický portál. 24. 11. 2008. [cit. 2013-12-17] Dostupný z WWW: < http://spomocnik.rvp.cz/clanek/11753/SITOVA-GENERACE-PODLE-TAPSCOTTA.html>.

Zajímavým vstupem do problematiky by mohla být výuka robotizace, která děti nenásilnou formou naučí algoritmickému myšlení. Děti dostanou k dispozici roboty, které se snaží dle instrukcí vyučujícího naprogramovat tak, aby plnili jejich požadavky, ale zároveň si procvičují i další dovednosti (tvořivost, pečlivost, přesnost a při skupinové výuce spolupráce).

Při výuce robotizace lze využít různé příběhy související s probíraným tématem. Jednou z možností je využití konstruktivistického přístupu, kde učitel poskytuje dětem tvořivou atmosféru, maximum praktických příkladů, které vedou k vyřešení daného problému, bez nutnosti jeho zásahu. Při sestavování robotů nejde jen o jejich programování, ale i o jejich zapojování, kde si děti procvičují jemnou motoriku. Celý systém začíná od sestavování robotů po tvoření programů na jejich řízení.[23]

Pro výuku robotizace lze využít robotické stavebnice například RoboKit nebo známější Lego Mindstorms. Zajímavou alternativou jsou Robotik od Merkuru, roboti od Conrada a mnoho dalších.

Další možností je zařazení propedeutického programovacího jazyka, ve kterém si žáci osvojí základy bez složitých konstrukcí. Při výběru vhodného jazyka je nutné brát v úvahu jazykovou lokalizaci a zvážit několik kritérií. Z pohledu logovské kultury se především jedná o jednoduché začátky, kdy každý žák by neměl mít problém v daném prostředí začít pracovat, aniž by měl předchozí znalosti s programováním. Z tohoto pohledu je možné nejčastěji zařadit programovací prostředí: Karel, Baltík, ImagineLOGO nebo Scratch. Všechna tato prostředí mají českou lokalizaci, tudíž jsou vhodná i do nižších ročníků.

Klasické programovací jazyky se svým charakterem spíše hodí pro využití ve vyšších ročnících, kde žáci mají již rozvinutější abstraktní a algoritmické myšlení. Možnosti volby jazyka jsou poměrně rozsáhlé. Při výběru jazyka je také důležité zvážit možnou návaznost výuky na střední škole. Vzhledem k malému počtu výzkumů v této oblasti, se lze částečně orientovat dle výzkumu[24], který popisuje, že nejvíce vyučovaným jazykem na SŠ je Object Pascal a Java, které oba mají 20% zastoupení. S 17 % je to C a Python. Mezi další

[23] TUNKROVÁ, Dana, HRBÁČEK, Jiří. *Výuka programování robotů na 1. Stupni ZŠ*. Veřejné služby IS. [cit. 2013-12-30] Dostupný z WWW: <https://is.muni.cz/repo/1078241/TunkrovaHrbacek.pdf>.

[24] KOTEK, Lukáš. *Jaké programovací jazyky se používají na středních školách?* [online]. Česká škola: 27. 5. 2013. [cit. 2014-01-01] Dostupný z WWW: <http://www.ceskaskola.cz/2013/05/lukas-kotek-jake-programovaci-jazyky-se.html>.

vyučované jazyky patří Basic, C# a PHP. Mezi doplňkovými jazyky se objevuje s 23 % PHP, s 18 % C/C++ a s 17 % JavaScript, což svědčí o velkém vlivu webových technologií. Z tohoto pohledu je možné se domnívat, že vhodnější volbou je procedurální programovací jazyk s možnou návazností výuky na střední škole.

Oblastí, která se v současné době těší velké popularitě mezi žáky vzhledem k její blízkosti k nim, jsou webové aplikace. Zároveň se jedná o oblast, která se neustále viditelně vyvíjí a se kterou je většina žáků v dennodenním kontaktu. Z tohoto důvodu školy často vyučují značkovací jazyk HTML včetně rozšíření o CSS a jeho další rozšíření v podobě programovacích jazyků JavaScript a PHP.

Před volbou, který jazyk vyučovat, je dobré zvážit, který z jazyků je vhodný pro danou školu a to z hlediska vlastností jazyka a z hlediska začlenitelnosti do ŠVP. Každý jazyk má svá specifika, tak jako každá škola. Zvolený programovací jazyk by měl u žáků rozvíjet algoritmické myšlení, jak požaduje RVP, a měl by být přínosem pro žáky v jejich budoucím životě. Z tohoto pohledu je při výběru jazyka nutné zvážit předchozí znalosti žáků, ať už s propedeutickým jazykem, či třeba jakým směrem se ubírala výuka dříve a na něj následně navázat, čímž se zvyšuje návaznost systému a motivovanost žáků. Věková hranice také není přesně dána, ale vzhledem k podstatě jazyků je vhodnější použití u žáků, kteří jsou starší a jsou schopni pochopit jednotlivé struktury a zákonitosti daného jazyka.

3.2.3. Programování a algoritmizace na ZŠ v Anglii

V Anglii je vzdělávání členěno do stupňů nazývaných Key Stage (KS), které se dají s českou školou zhruba srovnat takto. KS1 odpovídá ročníkům 1, 2. KS2 odpovídá ročníkům 3, 4, 5, 6. KS3 odpovídá ročníkům 7, 8, 9. KS4 by odpovídal ročníkům 10, 11, 12. Věkově je celý systém posunutý o dva roky. Děti začínají povinnou docházku v 5 letech a končí v 16 letech. Z tohoto pohledu lze konstatovat, že druhému stupni ZŠ by zhruba odpovídaly KS2, 3, 4.[25]

3.2.4. Právní rámec v Anglii

Programování a algoritmizace je v Anglii definována v národním kurikulu, které v současné době prochází rozsáhlým přezkoumáním a nová aktualizovaná verze by měla

[25] Department for education. *Stránky ministerstva školství Velké Británie* [online]. [cit. 2014-05-22]. Dostupné na World Wide Web:< http://www.education.gov.uk/get-into-teaching/subjects-age-groups/age-groups/graph-description>

platit od šk. roku 2014/2015. Do té doby platí staré kurikulum, které se opírá i o další dokumenty, které zohledňují rychlý vývoj v oblasti informačních technologií.[26]

Mezi cíle zahrnující oblast programování a algoritmizace pro oblast KS2 patří to, aby všichni žáci pochopili základní pojmy z informatiky, včetně abstrakce, logiky, algoritmizace a zápisu dat. Dále by měli umět analyzovat problémy při psaní počítačových programů, aby je bylo možno vyřešit. Národní kurikulum taktéž uvádí, co by se žáci měli v dané oblasti učit. V oblasti KS2 se žáci učí psát a ladit programy tak, aby dosáhli konkrétního stanovené cíle. Učí se řešit problémy tak, že je rozdělí na menší části a ty vyřeší. Učí se pracovat s proměnnými včetně jejich vstupů a výstupů. Programy rozdělují do sekvencí a učí se jednotlivé části opakovat. Učí se používat logické myšlení k vysvětlení, jak fungují jednoduché algoritmy a popřípadě najít chyby v algoritmech či programech a následně je opravit. [27]

Cíle pro oblast KS3 a KS4 jsou pro obě oblasti stejné a vychází z definice cílů celého kurikula, tudíž jsou stejné jako v oblasti KS2. Následně se jen liší, co se žáci v každé oblasti učí. Mezi základní učivo v oblasti, která by se dala zahrnout do algoritmizace a programování v KS3 patří modelování reálných problémů a jejich řešení. Důraz je kladen na pochopení klíčových algoritmů, které obsahují takzvané „počítačové myšlení“. Zároveň by se žáci měli učit navrhovat alternativní algoritmy, které vedou k řešení sejného problému alternativní cestou. Měli by umět používat dva programovací jazyky, ze kterých je alespoň jeden textový. Pomocí programovacího jazyka by měli řešit řadu výpočetních problémů a zároveň umět pracovat s datovými strukturami, jako jsou seznamy, tabulky nebo pole. Měli by navrhovat modulové programy, které obsahují funkce. Měli by chápat výrokovou logiku „a zároveň“, „nebo“, „negaci“ a používat je při programování. Zároveň se učí prezentovat čísla v binární soustavě a provádět s nimi základní operace.

[26] Department for education. *Stránky ministerstva školství Velké Británie ICT*]. 2013-08-02- [cit. 2014-01-11]. Dostupné na World Wide Web:
<http://www.education.gov.uk/schools/teachingandlearning/curriculum/primary/b00199028/ict>.

[27] Department for education. *Computing programmes of study: key stages 1 and 2*. 2013-09 [cit. 2014-01-11]. Dostupné na World Wide Web:
<http://www.computingatschool.org.uk/data/uploads/primary_national_curriculum_-_computing.pdf>.

Oblast KS4 z pohledu programování a algoritmizace obsahuje rozvoj kreativity a znalostí v oblasti výpočetní techniky. Dále by žáci měli rozvíjet a uplatnit své analytické myšlení při řešení problémů a být tvůrčí v oblasti „počítačového myšlení“.[28]

Jak z výše zmíněných cílů a učiva oblastí KS2, 3, 4 vyplívá, tak všichni žáci nám odpovídajícího druhého stupně prochází výukou programování dvou jazyků, ze kterých by měl být alespoň jeden textový. K výuce algoritmizace, respektive algoritmického myšlení na těchto školách, přispívá již výuka od mateřské školy, která v sobě obsahuje tyto prvky. Jejich podoba v počátcích je zaměřena na ovládání robůtků (např. podlahové želvy) nebo například klasické ovládání moderních přístrojů, které vyžaduje algoritmické myšlení.

3.2.5. Využívané možnosti algoritmizace a programování v KS3

Oblast KS3 je z hlediska programování v kurikulárním systému Anglie nejvíce zaměřená na tuto problematiku. Mezi důvody zaznívající pro programování zejména patří, že digitální gramotnost je čím dál tím důležitější v současné společnosti. Dalším důvodem je tvrzení, že podporuje duševní rozvoj k řešení problémů, se kterými se dá setkat i v jiných oblastech života.

Mezi nástroje, které jsou využívány k programování v Anglii, patří Alice, což je Open – Source prostředí, sloužící k výukovým účelům programování. Nástroj obsahuje knihovny obrázků a pozadí, které se dají využívat při jeho použití. Alice umožňuje žákům pochopení základů programování bez nutnosti znalosti zápisu kódu. Nástroj je zaměřen na vytváření událostí a metod, pomocí nichž si vytvářejí svůj vlastní virtuální svět s animovanými postavami. Výuka v tomto prostředí je didakticky zpracována do sedmi lekcí, které umožňují postupný vývoj žáka. V první lekci je žák zasvěcen do slovníku Alice, vstupů a výstupů. Ve druhé lekci se učí vytvářet nové metody. Třetí lekce přináší výuku událostí a postupné zdokonalování. Čtvrtá až šestá lekce přináší návrh, vývoj a hodnocení dětského projektu. V poslední sedmé lekci se zaměřují na prezentaci své práce.

Dalším využívaným nástrojem při výuce programování je Greenfoot. Jedná se o prostředí, které je zaměřeno na žáky, kteří již alespoň základním úvodem do programování prošli v rámci Scratch nebo Game Maker. Nástroj v sobě kombinuje propracovanost a sílu

[28] Department for education. *Computing programmes of study: key stages 2 and 3*. 2013-09 [cit. 2014-01-12]. Dostupné na World Wide Web: <http://www.computingatschool.org.uk/data/uploads/secondary_national_curriculum_-_computing.pdf>.

postavenou na komerčním jazyce Java s činnostmi založenými na vizuální dvoj rozměrné síti. Vývojové prostředí sice používá třídy, editor a kompilátor, ale je vhodné i pro začínající programátory. Výuka v tomto nástroji je doporučena minimálně na sedm hodin, ale v případě větších projektů se prodlužuje. Žáci začínají celou výuku úvodem do programování v jazyce Java, dále následuje práce s pohybujícími se objekty. Dalším fází při výuce je integrace a úprava obrázků souvisejících s programováním. Události jsou začleněny jakožto interaktivita závislá na kliknutí myši. Důraz je zde také kladen na design, který by měl být podobný hrám. V konečných fázích se žáci zaměřují na logiku celého svého projektu. Následně na jeho dokončení a vytvoření například celé hry.

Zajímavou a didakticky podporovanou možností výuky programování je například Visual Basic, který jako všechny výše zmíněné varianty je také k dispozici v Express verzi zdarma. Základní varianta výuky je opět sestavena na sedm hodin, které je možné se složitostí projektu libovolně rozšířit. V první fázi jsou žáci seznamováni se samotným prostředím, jak otevírat a spouštět programy a je jim ukázán ukázkový kód programu. Následuje práce s událostmi, při které si žáci procvičí možné události v programu a následné spuštění konkrétní reakce. V základní podobě se jedná například o stisknutí tlačítka a následnou změnu barvy pozadí, popředí, či změny velikosti tlačítek. V další fázi výuky se žáci učí programovací postupy, které vedou k rozdělení složitého problému na části méně složité. Žáci jsou v této fázi seznamováni s proměnnými a je jim vysvětlováno k čemu slouží. S těmito základními poznatky zkoušejí jednoduché matematické operace jako je sčítání či násobení. Po zvládnutí proměnných přichází na řadu větvení programu, kde je žákům vysvětleno k čemu větvení slouží a jak ho lze využít. K výuce jsou pro názornost používány i vývojové diagramy zaměřené právě na problematiku větvení. Žákův program by měl být na základě analýzy proměnné schopen určit následný postup. Předposlední fází je výuka různých druhů cyklů, kde se žáci učí výpisy proměnných a jejich změny při průběhu cyklem, popřípadě zkouší cyklus v cyklu. Výsledkem celé výuky v této oblasti může být návrh systému na příjem zakázek a rozvozu Pizzy, který bude nejprve teoreticky a designově spravován jako návrh a po schválení učitelem naprogramován v daném prostředí.

Zajímavou možností, kterou lze využít při výuce programování je nástroj Game Maker, který lze v základní verzi či s použitím přes webové rozhraní taktéž pořídit zdarma. Zde se žáci během několika málo hodin naučí nástroj a jeho základní funkce ovládat tak, aby byli

schopni navrhnout 2D či 3D hru. Základní ovládání programu lze realizovat pomocí drag a drop a pro pokročilejší možnosti nástroj obsahuje vlastní programovací jazyk. Při základním použití nástroje není ani pro začátečníky problém vložit vlastní pozadí hry, vytvořit animaci, či vložit zvuky včetně efektů. V případě, že si žáci zakoupí plnou verzi nástroje, je možné vlastní vytvořené hry prodávat. Celá výuka je opět členěna do několika celků, které se zaměřují na klíčové aktivity. Ty tentokrát na rozdíl od předcházejících aktivit jsou spíše zaměřeny do oblastí života, jako je: design, marketing, řízení prodeje, obchodování a další. Tvorba předem naplánovaných aktivit umožňuje žákům vytvářet program v přesně po sobě jdoucích krocích, které přispívají k rozvoji algoritmického myšlení.[29]

Jak vyplývá z uvedených příkladů, které jsou pouze malou částí z množství podpůrných materiálů pro výuku programování v Anglii, je této problematice věnován poměrně velký prostor. V celém systému lze najít mnoho metodických podpor pro učitele, které lze velmi snadno zavést do výuky díky jejich metodickému zpracování.

3.2.6. Programování a algoritmizace na ZŠ ve Slovenské republice

Členění základní školy na Slovensku je obdobné jako v naší republice. Základní škola je dle klasifikace UNESCO rozdělena do dvou stupňů ISCED1 odpovídající prvnímu stupni a ISCED2 odpovídající našemu druhému stupni. Na obou stupních je na školách vyučována problematika základů programování a algoritmizace v rámci vzdělávací oblasti „Matematika a práce s informacemi“. Vzhledem k propojenosti programů prvního a druhého stupně budou v další kapitole zmíněna i témata prvního stupně.

3.2.7. Právní rámec na Slovensku

Problematika algoritmizace a programování na Slovensku vychází ze státního vzdělávacího programu, kde je začleněna v oblasti „Matematika a práce s informacemi“. Předmět, který problematiku pokrývá, je nazván „Informatická výchova“ na stupni ISCED1 a „Informatika“ na stupni ISCED2. Předmět jako takový je v celém kurikulu považován za velmi důležitý, protože v žácích rozvijí myšlení, schopnost analýzy, syntézy a hledání vhodných strategií, které se dají ověřit v praxi.

[29] Computing at school. *Computer Programming in Key Stage 3*. 2009-09 [cit. 2014-02-11]. Dostupné na World Wide Web: <http://www.computingatschool.org.uk/data/uploads/CPinKS3.pdf>.

V rámci předmětu „Informatická výchova“ je na stupni ISCED1 zařazeno pět okruhů, kterým se žáci v rámci předmětu věnují. Jeden z okruhů se nazývá „Postupy, řešení problémů, algoritmické myšlení“ a je zaměřen právě na počátky výuky programování a algoritmizace. V tomto okruhu se žáci věnují různým způsobům řešení problémů pomocí informačních technologií. Učí se základním pojmům jako je algoritmus, algoritmizace, programování, program. Žáci se nejen učí algoritmickému myšlení a řešení problémů z oblasti informačních technologií, ale i z oblasti praktického života, kde se dají tyto postupy uplatnit.

Mezi základní postupy a metody zařazené do výuky patří skládání různých stavebnic a hlavolamů, které se učí postupně sestavit či vyřešit dle návodů. Učí se řešit jednoduché algoritmy, které kreslí obrázky, či vytváří animace pomocí propedeutického programovacího jazyka. Dále se učí řadit jednotlivé příkazy v logickém sledu za sebou, skládat obrázky z menších obrázků, či řídit robota v počítačovém prostředí. Učí se vytvářet návody a postupy nebo podle nich v přesně stanovených krocích postupovat a plnit je.[30]

Na stupni ISCED2 je vyučován v rámci oblasti „Matematika a práce s informacemi“ předmět s názvem „Informatika“, který zahrnuje problematiku algoritmizace a programování. Mezi jeden z hlavních cílů na tomto stupni patří objasnění pojmů a technik používaných při práci s daty a při tvorbě algoritmů. Část vzdělávacího procesu předmětu „Informatika“ je zaměřena k porozumění pojmům program a algoritmizace se zaměřením na formální zápis programu pro automatické zpracování údajů. V další části si žáci ještě více rozvijí schopnost algoritmizace praktických problémů a rozvijí si programátorské dovednosti. Stejně jako na stupni ISCED1 se zde nachází pět okruhů, kde opět „Postupy, řešení problémů, algoritmické myšlení“ nejvíce rozvijí algoritmizaci a programování.

Mezi základní postupy výuky v této oblasti patří zaměření na programovací jazyk, ve kterém se učí základní příkazy. Zavádí pojem proměnná a učí se do ní přiřazovat hodnotu. Vytváří správnou posloupnosti příkazů, na kterou navazuje výuka procedur a různých cyklů. Zjišťují různá časová řešení konkrétních problémů. Na konci tohoto období by žák měl umět formálně zapisovat navržené postupy, měl by umět vytvořit algoritmus na zašifrování jednoduchého textu. Měl by umět v propedeutickém programovacím jazyce

[30] *Štátny vzdělávací program. Informatická výchova.* Štátny pedagogický ústav, Bratislava:2008. [cit. 2013-12-08] Dostupný z WWW: < http://www.statpedu.sk/files/documents/svp/1stzs/isced1/vzdelavacie_oblasti/informaticka_vychova_isced1.pdf>.

přiřazovat své výpočty do proměnných, seskupovat vhodné části programu do procedur a řešit úlohy pomocí opakování určité části programu. Měl by umět porovnávat časy, které jsou potřeba k vyřešení problému různými cestami.[31]

3.2.8. Využívané možnosti programování a algoritmizace na Slovensku

Jak z právního rámce na Slovensku vyplývá, vydalo se taktéž Slovensko cestou podpory výuky programování a algoritmizace již na základních školách. Oporou těmto školám může být bratislavská Univerzita Komenského, která se touto problematikou zabývá a poskytuje učitelům velmi dobrou metodickou podporu. Dále se snaží vyvíjet své propedeutické programovací jazyky, které přispívají k větší podpoře výuky programování.
Mezi rozšířené programovací jazyky, které se na Slovensku vyučují, patří Imagine Logo. Učitelé a žáci mají k dispozici učebnice a podpůrné materiály, které jsou určeny pro druhý stupeň ZŠ. Výuka je rozčleněna do několika oblastí. Nejprve se žáci v programovacím jazyce učí malovat, učí se volit jednotlivé barvy a směry pohybů. Po zvládnutí základů malování se učí některé kroky, které již vykonali opakovat nebo je pojmenovat a kdykoliv v budoucnu je znovu použít. Do výuky začleňují i matematické dovednosti z oblasti geometrie, které využívají při malování geometrických obrazců. Učí se měnit pozici, ze které kreslí, aby obrazce nemusely být spojité. Při zásahu uživatele je vyvolávána reakce, což je úvod do událostí. V průběhu vývoje výuky jsou přidávány další objekty, které mají své vlastnosti a mohou být ztvárněny i animovaně. V závěrečné fázi výuky přichází na řadu proměnné, do kterých si žáci ukládají potřebné hodnoty pro další zpracování. Výsledkem jejich výuky mohou být jednoduché hry, které simulují oblíbené stolní hry.[32]
Úspěšným předchůdcem programu Imagine Logo byl Comenius Logo, který měl taktéž dobrou metodickou podporu ve formě učebnic a propracovaných metodických materiálů pro samotné učitele. Vzhledem ke stáří programu již není podporován na nejnovějších operačních systémech řady Windows. Program taktéž pracuje s objektem nebo více objety ve formě želvy. Programovací jazyk umožňuje kreslení různými čarami a barvami.

[31] *Štátny vzdělávací program. Informatika.* Štátny pedagogický ústav, Bratislava:2008. [cit. 2013-12-08] Dostupný z WWW: < http://www.statpedu.sk/files/documents/svp/2stzs/isced2/vzdelavacie_oblasti/informatika_isced2.pdf>.
[32] BLAHO, KALAŠ. *Tvorivá informatika. 1. zošit z programovania*. Vydání 3. Bratislava: SPN, 2009. 48 s. ISBN 978-80-10-01723-2.

Samozřejmostí je tvorba procedur a možnost zápisu příkazů pomocí jazyka Logo, který bývá přeložen do lokalizované verze.[33]

Další poměrně novou variantou jazyka Logo je EasyLogo, které se dá pořídit i v české verzi zdarma. Vývoj tohoto prostředí byl zaměřen i na uživatele, kteří nemají rozsáhlé počítačové dovednosti. Z tohoto důvodu vznikla verze, která se snaží o zjednodušení celého prostředí a programování například oproti verzi Imagine Logo. Prostředí je rozděleno do 2D mřížky, ve které se dané objety mohou pohybovat. Otočení a pohyb objektu je zde možný o 45° a po pohybu jsou objekty přitahovány k mřížce. V jazyce Logo zůstalo jen pár možností, mezi které patří příkazy pro kreslení, cykly a procedury. Programovací prostředí obsahuje aktivity, které jsou zaměřeny na výuku programování konstruktivistickým přístupem. Program lze ovládat bez klávesnice pouze pomocí myši pomocí přetahování kartiček. Sada připravených úkolů umožní žákům seznámení se s celým prostředím nebo umožní učiteli dle nich realizovat výuku. Jednotlivé úkoly lze libovolně přeskakovat, prostředí nijak nekontroluje jejich plnění. Mezi první cíle patří základní ovládání objektů (mohou mít různou podobu) pomocí klikání na tlačítka. Byť na první pohled tato aktivita vypadá snadně, lze různými kombinacemi pohybů dosáhnout poměrně složitých úkolů, které vyžadují algoritmické myšlení. Pokud žák zvládá tyto pohyby, přechází k další fázi, kdy tyto pohyby sestaví do sekvence příkazů. Tím vytvoří první program, který obsahuje několik příkazů za sebou. Zadáním programu může být například úkol, ve kterém se objekt pohybuje po předem připravených místech a musí je všechna navštívit. Dalším krokem je výuka tvorby cyklů, která se hodí k opakování činnosti, kterou objekt vykonává. Úkolem může být opakovaný sběr předmětů, které jsou v nějakém pravidelném sledu uspořádány za sebou. Pokud žáci zvládají základy opakování části programu, přechází na složitější vzory, ve kterých musí objevit opakující se část. Následně navrhnou algoritmus opakování a vloží ho do programovacího prostředí. Složitějším úkolem je tvorba procedur, které kreslí různé tvary. Po zvládnutí této části mohou žáci nechat nakreslit nadefinovaný tvar na určeném místě a sestavit třeba domeček, který se skládá z obdélníku a trojúhelníku. Posledními aktivity je zaměření se na opravy či

[33] BLAHO, KALAŠ. *Comenius Logo: tvorivá informatika*. Bratislava: CL Group, 1998. 157s. ISBN 80-967999-0-8

možná vylepšení programu, která by například zkrátila zápis nebo našla nedostatky v algoritmu.[34]

Možností výuky algoritmizace a programování na ZŠ není pouze koncepce postavená na jazyku Logo, ale i například na jazyku Pascal. Pro žáky a učitele je k dispozici podpora ve formě učebnice doplněné o knihovny, kterou lze využít na druhém stupni ZŠ nebo na středních školách. Výuka probíhá formu řešení příkladů, na kterých je žákům vysvětlována teorie. Jelikož se jedná o klasický programovací jazyk, je i postup výuky řešen v obvyklém sledu. Žáci si prochází lekcemi, kde postupně probírají proměnné, cykly, procedury, vstupy, podmínky a funkce. Učí se pracovat s grafikou a zpracovávat vstupy, které přicházejí buď z klávesnice, nebo od myši.[35]

3.2.9. Programování a algoritmizace na ZŠ v Austrálii

Jednou z oblastí, která je v systému vzdělávání velice dobře hodnocena, je Austrálie. Vzdělávací systém se sice v jednotlivých zemích liší, ale nejedná se o diametrální rozdíly. Děti mají povinnou školní docházku 10 let, což je přibližně do věku 15 – 16 let. Vzdělávání v oblasti programování a algoritmizace je zařazeno do předmětu „Digitální technologie“. Předmět je vyučován od prvního ročníku a však pojem algoritmizace a programování se objevuje až v ročníku sedmém.

3.2.10. Právní rámec programování v Austrálii

V sedmém a osmém ročníku se žáci učí posuzovat a konstruovat algoritmy. Při posuzování předpovídají na základě vstupní hodnoty, hodnotu výstupní. Při tvorbě algoritmů se zaměřují na tvorbu takových, které umí například třídit a hledat. Před použitím algoritmu se pokouší analyzovat jeho správnost a to na základě zadaných vstupních údajů s předpokládaným výstupem. Navržené algoritmy zapisují do vývojových diagramů, které následně přepisují do výpočetních instrukcí. Strukturovaně vytváří algoritmické pokyny v angličtině (například do-while), které umožňují tvorbu cyklů. V oblasti programování je v sedmém a osmém ročníku zařazeno vytváření a úprava programů, které umožňují větvení, cykly, funkce. Do programů jsou zaváděny algoritmy, které byly připraveny a

[34] SALANCI, Ľubomír: *EasyLogo-discovering basic programming concepts in a constructive manner. In: Constructionist approaches to creative learning, thinking and education: Lessons for the 21st century.* Bratislava: FMFI UK, 2010. ISBN 978-80-89186-66-2, ISBN 978-80-89186-65-5. [cit. 2014-02-16] Dostupný z WWW: <http://www.salanci.sk/EasyLogo/Paper.pdf>.

[35] BELUŠOVÁ, VARGA, ZIMANOVÁ. *Algoritmy s Pascalom*. Bratislava: SPN, 2002. 48 s.

učeny v rámci oblasti algoritmizace. Součástí programování je například tvorba her, které umí manipulovat s objekty. Dále naprogramování robota, který objekty pozná a manipuluje s nimi na základě barvy. Zkouší vytvářet informační kiosek, který má několik vrstev tlačítek a má zpětnovazebný systém s jeho uživatelem. V devátém a desátém ročníku je algoritmizace zaměřená na řešení reálných problémů se začleněním prvků start, end, if, do. Opět jako v nižších ročnících je zde analýza algoritmů s odhadem předpokládaného výstupu. Zaměření této oblasti je také na tvorbu algoritmů, které žáci vytvářejí podle zadávací dokumentace a následné ověření funkčnosti při nekorektních vstupech. V oblasti zaměřené na programování se žáci propracují i k objektově orientovanému programovaní. Následně programují jednotlivé moduly, ze kterých se dá vytvořit funkční program, který splňuje zadané požadavky. Učí se definovat třídy, vlastnosti objektů. Zabývají se různými datovými strukturami například typu pole a řeší vhodným způsobem a algoritmem konkrétní problém. Z kurikula vyplývá, že programování v Austrálii je vyučováno až ve vyšších ročních, ale s poměrně velkou obsahovou náplní.[36]

3.2.11. Komparace uvedených kurikulárních systémů

Při pohledu na čtyři zmíněné systémy, lze dospět k názoru, že programování a algoritmizace je nejméně zastoupena v systému české republiky, kde je sice vyučován předmět zaměřený na problematiku ICT s doporučenou jednou hodinou na druhém stupni, ale z kurikulárních dokumentů nevyplývá povinnost škol výuku algoritmizace či programování do výuky začlenit. Na druhou stranu školy, které mají zájem tuto problematiku vyučovat, mohou z disponibilní hodinové dotace navýšit množství hodin věnovaných této problematice. Pokud tak učiní, lze z klíčových kompetencí a cílů oblasti „Informační a komunikační technologie“ vybrat formulace, které jim výuku této problematiky umožní.

Oproti českému systému je v Anglii problematice věnováno mnohem více času a děti se v základní podobě s algoritmizací setkávají již ve školce. Ve věkové kategorii našeho druhého stupně se nejprve učí algoritmizovat, rozdělovat větší úkoly na menší a volit správný postup řešení. Učí se problematice proměnných a datových struktur. Ve vyšších

[36] Australian Curriculum, Assessment and Reporting Authority, *Digital Technologies* [online]. [cit. 2014-02-19]. Dostupné na World Wide Web: < http://www.australiancurriculum.edu.au/technologies/digital-technologies/Curriculum/F-10>.

ročnících že žáci učí výrokovou logiku, funkce, větvení programů, práci s moduly. Ve věkové kategorii odpovídající našemu 6. ročníku žáci pracují se dvěma programovacími jazyky, kde jeden je textový a druhý může být propedeutický. Při výuce programování zároveň analyzují stávající algoritmy a učí se navrhovat jejich možná zlepšení a využití. Při hodnocení tohoto systému, lze dospět k názoru, že směrem k výuce programování a algoritmizace je z uvedených nejpropracovanější a této problematice se věnuje nejvíce ze všech systémů. Z pohledu přípravy na budoucí život jsou po úspěšném absolvování žáci schopni pracovat s moderními technologiemi a mají základy „počítačového myšlení". Někteří z nich mají otevřenější cestu stát se například programátory, kterých dle výzkumů bude v budoucnu nedostatek a tím podpořit tamní ekonomiku.

Zajímavou cestou se také vydala Slovenská republika, která se snaží taktéž žákům již od prvního stupně poskytnout základy algoritmického myšlení. Celý systém je spíše než na klasickém programovaní, postaven na „počítačovém myšlení", které všem žákům otevírá pomyslné dveře do života, ve kterém jsou všudypřítomné technologie. I za těchto podmínek je na školách vyučována problematika proměnných, procedur, cyklů a logického sledu programu. Slovensko sice této problematice nemá ve výuce věnováno tolik prostoru jako Anglie, ale jedná se o správný směr, který je otevřen budoucímu vývoji.

Posledním z uvedených systémů byla Austrálie, která je velmi vyhledávaným cílem studentů, byť vyšší věkové kategorie. Při výuce žáků odpovídající věku naší ZŠ, mají dvě oblasti, na které se zaměřují. Jednak je to oblast algoritmizace a tou druhou je oblast přímo programování. Obě oblasti na sebe navazují tak, že nejprve probíhá výuka algoritmizace a na ni navazuje programování. Celá výuka programování je velmi propracovaná opět od proměnných, přes funkce, cykly, větvení až po tvorbu modulů, které se dají využít při projektovém vyučování. V porovnání s ostatními systémy je velmi podobný vzdělávání v Anglii, ve které je této problematice na kurikulární úrovni věnován větší prostor.

3.3. Koncepce výuky základů programování na ZŠ

Využití programování se v dnešní době stává nedílnou součástí znalostí dnešních dětí, které potřebují doupravit své webové prezentace, pozměnit skript na stránce nebo se chtějí programováním v budoucnu profesionálně živit. Z tohoto důvodu je vhodné přemýšlet o správné koncepci výuky programování. Koncepce výuky programování je jednou z možností, jak žáky více motivovat k programování, jak žákům poskytnout větší podporu

při výuce, jak v žácích nalézt zájem a třeba i talent v této oblasti, ale i jak celou problematiku žákům znechutit. Z tohoto pohledu je důležité nalézt vhodnou koncepci, která přispěje k výuce v pozitivní rovině. Každá koncepce je souborem cílů, metod, organizačních forem, prostředků a dalších aspektů, které při správném sladění mohou vytvořit vhodné podmínky pro kvalitní vzdělávání. Při návrhu koncepce cíle většinou známe, je však důležité vhodně zvolit organizační formy, metody a využít dostupné didaktické prostředky, které jsou na školách na různých úrovních.

3.3.1. Organizační formy výuky

Mezi základní organizační formy, které se z jejich velkého množství a kombinací vyčlenily, lze podle (Skalková, 2007)[37] zařadit frontální vyučování, které probíhá v klasické vyučovací hodině a při výuce programování ve škole ho lze zařadit. Výhodou je práce s celou třídou, která probíhá v přesně ohraničeném čase a s konkrétní skupinou žáků. Při těchto hodinách lze plnit konkrétní dílčí či celkové cíle a zároveň má učitel kontakt s celou třídou. I při této formě je možné k žákům zachovávat individuální přístup a sledovat jejich individuální rozvoj a průběžně přejít v rámci hodiny na jinou organizační formu.

Frontální vyučování může být sice individuální, ale dochází pouze k interakci mezi učitelem a žákem. Žák je z tohoto pohledu jakoby vytržen ze své sociální skupiny, které je nedílnou součástí. Pokud se pokusíme žákovu roli zachovat a vytvořit tak pro něj příznivé prostředí, je další možnou formou skupinové a kooperativní vyučování. V malých skupinách po cca 4 žácích dochází k plnění zadaných úkolů. Výhodou těchto skupin je možnost individuálního zapojení i slabších žáků do řešení problému a jejich sociální interakce s ostatními. Skupinové práce se více hodí v situacích se složitějšími úkoly, či při řešení problému. Při skupinové práci lze uplatňovat kooperativní nebo kolaborativní přístup, dle našich cílů a složení skupin.

Další využívanou formou se stala individualizace a diferenciace v procesu vyučování. Základním principem individualizace je nalezení vlastního, optimálního přístupu k učení a vzdělávání pro každého žáka, jež odpovídá jemu přiměřené námaze. Případem snahy o individualizaci je například daltonský plán. Základní forma diferenciace vede k rozdělení třídy do skupin dle intelektových předpokladů, avšak tento přístup je kritizován za svou

[37] SKALKOVÁ, J. *Obecná didaktika*. Praha: Grada, 2007. ISBN 978-80-247-1821-7.

selektivnost. Z tohoto důvodu dochází ke kombinaci dalších postupů například v rámci frontálního vyučování se zařazením prvků problémového vyučování.

Organizačních forem existuje velká řada a jejich dělení je možné z mnoha hledisek, ale to není cílem této práce. Důležitější je zamyšlení nad možnostmi, které zmíněné formy přináší a které se dají v koncepci programování využít.

3.3.2. Vyučovací metody

Vhodnou volbou vyučovací metody lze žáky motivovat k procesu učení se a celý proces jim usnadnit. Z tohoto důvodu je i při výuce programování o zvolených metodách nutno přemýšlet a to především ve směru k žákovi. Správně zvolená metoda vede k dosažení stanovených cílů rychleji a efektivněji. V současné době existuje velká řada metod a učitelé si mohou z jejich pohledu vybrat právě tu nejvhodnější, která povede k efektivnímu naplnění cílů, ale určitě ne všechny jsou vhodné pro výuku algoritmizace a programování.

Metody lze klasifikovat dle velké řady kritérií, což není cílem této práce, ale existuje velká řada základních metod, které při výuce programování a algoritmizace využít lze. Mezi základní využívané metody patří metody slovní, které jsou nejen hojně využívané, ale zároveň doplňují i další metody, které by bez nich ztrácely význam. Slovní metody lze rozdělit na metody monologické, mezi něž patří vyprávění, vysvětlování a školní přednáška. Druhou skupinou jsou metody dialogické, které umožňují vzájemnou komunikaci mezi žáky a učitelem. Při dialogických metodách dochází k aktivizaci žáků vedoucí k rozvoji kognitivních schopností. Dialogické metody lze rozdělit na metody rozhovoru, dialogu, diskuze a brainstormingu. Poslední skupinou jsou metody práce s textem, které mohou být v podobě knihy, učebnice, elektronického textu. Tyto metody umožňují samostatné studium, či upevňování již získaných poznatků[38].

Další velkou skupinou metod jsou metody názorně demonstrační, vedoucí ke konkrétnímu poznání skutečnosti s možností přímého styku s vyučovanou realitou, mezi níž může patřit tvorba konkrétních programů. Do těchto metod lze zařadit pozorování předmětů a jevů, předvádění (předmětů, činností, pokusů, modelů), demonstraci statických obrazů a statickou a dynamickou projekci.[39] Právě předvádění činností lze v hodinách algoritmizace a programování využít ke zvýšení motivace, neboť správně zvolené demonstrační metody dokáží žáky velmi motivovat a probudit zájem o probíranou látku.

[38] MAŇÁK, J., ŠVEC, V. *Výukové metody*. Brno: PedF MU, 2003. ISNB 80-7315-039-5.
[39] MOJŽÍŠEK, L. *Vyučovací metody*. 3. vydání, Praha: SPN 1988

Poslední velkou skupinou metod jsou praktické činnosti, které umožňují žákům konkrétní praktické činnosti reálně vykonávat. I tyto metody lze rozdělit na didaktické montážní a demontážní práce, laboratorní práce, praktické pracovní činnosti a práce žáků různého obsahového zaměření. Právě práce žáků různého obsahového zaměření lze využít při tvorbě programu, který může být celý či částí praktického výstupu z hodin programování[40]. Jak je z výše uvedeného textu patrné, množství metod a jejich variací je poměrně velké. Záleží na každém učiteli, kterou vybere a která se bude hodit do jeho koncepce.

3.3.3. Didaktické prostředky

Pro praktickou výuku programování jsou nezbytnou součástí vhodné didaktické prostředky zejména z IT oblasti. Sice lze vyučovat programování bez IT techniky, což dokazují „Unplugged" aktivity, ale pro praktické využití s nácvikem konkrétního prostředí nepostačují. Navíc velké množství těchto aktivit je právě k dispozici na webových stránkách.

Pro výuku algoritmizace jako takové, bez návaznosti na programování, se lze bez IT techniky obejít, avšak její použití pro tuto oblast je také vhodné.

Didaktickými prostředky se rozumí materiální předměty (i software), které zefektivňují vyučovací proces. Pro výuku programování je důležitá část prostředků, nazývaná didaktická technika. Pro výuku programování je zapotřebí taková technika, která umožní využití zvoleného programovacího jazyka. V současné době tomuto požadavku nejvíce vyhovuje PC, avšak část práce lze přesunout na smartphone nebo tablety. Zefektivnění celého procesu lze dosáhnout pomocí internetu, který umožňuje výukový obsah zobrazit odkudkoliv a kdykoliv.

3.3.4. Klasifikace koncepcí

Vzdělávání v oblasti programování a algoritmizace lez rozčlenit na školní (formální) a neškolní (neformální, informální). Školním vzděláváním lze rozumět výuku ve škole, která má svá specifika a pravidla. Neškolním vzděláváním lze rozumět dobrovolné vzdělávání ať už organizované či ne, kterého se vzdělávaný sám účastní a má z nějakého důvodu silnější motivaci se danou problematiku naučit. Každá skupina má svá specifika.

[40] MAŇÁK, J. *Nárys didaktiky*. 3. Vydání, Brno: PedF MU, 2003. ISBN 80-210-3123-9

Většina koncepcí ve školách je postavena na učiteli, který se vhodnými metodami a formami snaží předat žákovi danou problematiku. Většinou se jedná o koncepce, které nevyužívají pro výuku jako takovou, elektronickou podporu nebo nabízí pouze texty v elektronické podobě. Dále tyto koncepce budou nazývány „klasické koncepce“.
Na rozdíl od škol, si zájemci při neorganizovaném neškolním vzdělávání informace vyhledávají často sami a při tom využívají možnosti elektronické podpory. Tím se rozumí například možnost vyzkoušení části programu a jeho reálné ověření. Ve školách se tyto koncepce vyskytují v menším zastoupení. Tyto koncepce budou dále nazývány „koncepce využívající elektronickou podporu“.

3.3.5. Klasické koncepce

Klasické koncepce jsou často využívány ve školním prostředí, kde převážná část předávání informací stojí na učiteli, který je schopen danou problematiku individualizovaně předat dle potřeb konkrétního žáka.
Do klasických koncepcí se dají zařadit knihy, vysvětlující problematiku algoritmizace či programování. Knihy lze využít pro samostudium, což není úplná koncepce výuky, ale spíše se dají použít jako vodítko, kterého se učitel drží a ze kterého například žáci vypracovávají úlohy a které jim poskytuje učební podporu.
Další klasickou koncepcí je výuka podle metodicky zpracované příručky, která pomáhá učiteli krok za krokem celou problematiku vysvětlit a je doplněna příklady. V tomto případě může jít i o výuku bez metodicky zpracované podpory, která je pouze didakticky zpracována učitelem s konkrétními úlohami na procvičení.
Koncepcí, stejně tak jako metod a forem je velká řada a dost často nové vznikají právě jejich kombinací. Jak bylo uvedeno v kapitole Programování a algoritmizace na ZŠ v ČR a vybraných státech světa je problematika programování, ať už se jedná o klasické či propedeutické jazyky, většinou didakticky řazena stejně až na malé výjimky. Z tohoto důvodu bude z této posloupnosti vycházeno i v dalších kapitolách, které se posloupností a didaktickou stránkou jednotlivých koncepcí zabývají.

3.3.5.1. Knihy o programování

Knih o programování je samozřejmě velká řada a každá z nich má jednotlivá témata zpracována dle potřeb daného jazyka, ale prostudováním několika z nich lze dospět k závěrům, že zde platí určité postupy, které se taktéž prolínaly výše zmíněnými kapitolami, které se věnovaly příkladům programování v různých státech světa.

Pro potřeby výuky programování na základní škole je vhodné vybrat knihu, která se bude zabývat programovacím jazykem od začátku, což umožní žákům vstup do problematiky a jejich postupný vývoj. Ve zbytku této kapitoly budou uvedeny posloupnosti programovacích celků vycházející z různých knih o programování. Pořadí těchto celků může být v různých jazycích a formách knih odlišné, ale i mezi nimi lze sledovat souvislosti.

Bez odlišností, které se týkají vzhledu, ovládání a instalace konkrétních programovacích prostředí, většina úvodů do programování začíná výpisem nějakého textu typu „Hello World“, ve kterém postupně přibývají další slova, odřádkování a různá spojování textů.

Další krok je věnován problematice proměnných, kde v některých jazycích dochází nejprve k vysvětlování právě jejich deklarace a u některých ne neboť není potřebná. Pak přichází na řadu pojmenování proměnných, které až na malé odlišnosti má stanovená podobná pravidla. Do proměnných jsou přiřazována čísla, řetězce a v konečné fázi bývá vysvětlováno pole či více rozměrné pole. Při ukládání dat do proměnných je vysvětlováno jak uložit proměnnou do proměnné, matematické operace s proměnnými a textové operace s proměnnými (práce s operátory).

Po zvládnutí proměnných bývá zařazen celek týkající se podprogramů, které bývají nazývány funkcemi či procedurami a vysvětlení jejich rozdílu. Bývá zde vysvětlen rozdíl mezi podprogramem s argumentem a podprogramem bez argumentu, jaký je mezi nimi rozdíl a jaké mohou být rozdíly ve výstupech.

Po podprogramech přichází na řadu větvení programů, které se nejprve zabývá tím, co jsou podmínkové operátory a jak se dají využít při větvení programu. Následují vnořené podmínky a logické operátory.

Velkým celkem, který následuje, jsou cykly, umožňující opakování části či celý program. Ve většině jazyků se jedná o cykly s podmínkou na začátku, s podmínkou na konci či cykly typu „for“. Je vysvětlováno, kdy je vhodné použít konkrétní cyklus a kdy ne.

Další části jsou spíše specifické konkrétním prostředím. Je zde uváděno u objektově orientovaného programování, co jsou to objekty, třídy, jaké mají vlastnosti a jak se s nimi dá pracovat. Zajímavou oblastí jsou události, které jsou pro různá prostředí rozdílné. Bývá zde uváděna problematika vstupů a výstupů a jejich specifika. Pokud jazyk umožňuje práci s grafikou, tak zde bývají kapitoly věnovány této problematice. Většina koncových kapitol bývá zaměřena na problematiku ladění programů, vyhledávání chyb či optimalizaci programů.[41],[42],[43]

3.3.5.2. Klasická koncepce s příklady

Klasické koncepce s příklady často vycházejí z podobných posloupností, které byly uvedeny v předcházející kapitole. Na rozdíl od knih, jsou většinou konkrétně zpracovány pro určitou věkovou kategorii a doplněny vhodnými příklady. V této kapitole bude prozkoumána konkrétní koncepce výuky v prostředí Delphi na ZŠ.

Koncepce je určena žákům osmých či devátých ročníků na půl roku s dvojhodinovou dotací týdně. Koncepce se skládá ze sylabu a výukového materiálu. Sylabus uvádí, jaká problematika musí být v základních tématech probrána. Výukový materiál je určen pro žáky a učitele. Je koncipován jako příručka, která obsahuje vysvětlení a procvičení problematiky na konkrétních příkladech. U příkladů jsou návrhy správného řešení.

Při výuce je důraz kladen na výuku učitelem nikoliv na samostudium žáky. Probírané učivo by mělo být učitelem rozděleno do adekvátních celků a nemělo by být předkládáno čistě z výukového materiálu. Každá hodina by měla mít klasické fáze (motivační, expoziční, fixační a diagnostickou).

Celá koncepce je rozdělena na základy, řízení chodu programu, pole a řetězce a na podprogramy. V základech je probírán základní výstup programu („Hello World"). Proměnné a jejich deklarace a přiřazovací příkaz. Celý výklad je doplněn průběžnými příklady. Na konci tohoto celku je zadáno 6 otevřených úloh, které by se měl žák pokusit vyřešit pomocí vlastního programu.

Řízení chodu programu je část věnovaná především větvení programu a cyklům. Žáci jsou zde seznámeni s podmínkami a jejich syntaxí. Jsou zde probírány podmínkové a logické

[41] SATRAPA, P. *Perl pro zelenáče*. Praha, Neokortex 2000, ISBN 80-86330-02-8.
[42] KADLEC, V. *Učíme se programovat v Delphi a jazyce Object Pascal.* Praha, Computer Press 2001, ISBN 80-7226-245-9.
[43] PÍSEK, S. *JavaScript efektní nástroj oživení WWW stránek.* Praha, Grada 2001, ISBN 80-247-0014-X.

operátory. Následují cykly s podmínkou na začátku, s podmínkou na konci a cyklus „for“. Celou kapitolou se prolínají průběžné příklady.

Třetí část je věnována polím a řetězcům. Je zde vysvětleno co je pole, jak se do něj ukládají data a jak se k nim zpětně přistupuje. Při práci s řetězci je část věnována funkcím, které umí pracovat s řetězci a dokáží zjišťovat potřebné hodnoty. I touto částí se prolínají příklady na procvičování problematiky.

Poslední část je věnována podprogramům. Je zde vysvětleno, co je podprogram, druhy podprogramů a práce s parametry procedur. Poslední čím se příručka zabývá, jsou funkce. Na konci celé příručky je uvedeno možné správné řešení příkladů.[44]

3.3.6. Koncepce využívající elektronickou podporu

Mezi koncepcemi využívající elektronickou podporu patří zejména prostředí související s oblastí programování webových aplikací, které ke svému běhu využívají velkou řadu programovacích nástrojů. Jedná se především o jazyky a technologie HTML, CSS, JavaScript, VBScript, Java a různá SQL.

Základní koncepcí spadajícím do této kategorie jsou metodicky spravované příručky, které mají v elektronické podobě (většinou na webových stránkách) vypracovanou podporu, která často obsahuje příklady a úlohy. Druhou koncepcí je čistě elektronická podpora často ve formě webové stránky, sloužící spíše jako referenční příručka. Ovšem velkou výhodou těchto koncepcí je, že mají často možnost osvojení a vyzkoušení si části programů bez nutnosti instalace konkrétního programovacího prostředí. Druhá zmíněná koncepce je spíše vhodná pro samostudium, ke kterému musí být studující vnitřně motivován.

3.3.6.1. Metodické příručky s elektronickou podporou

Metodické příručky s elektronikou podporou stejně jako většina koncepcí bez elektronické podpory vychází z podobné posloupnosti výuky. Rozdílem je propracovanější a hlavně dostupnější studijní a podpůrný systém, který se nachází v elektronické podobě a je pro většinu žáků dostupnější při jeho vhodném umístění například na webových stránkách. Koncepce, která zde bude dále popsána, je zpracována právě tímto způsobem a je určena pro výuku programování Visual Basic.

[44] DOHNAL, P. *Programování – Delphi 7 Object Pascal*. Masarykova Univerzita, Brno 2009.

Výuka touto koncepcí je rozdělena na algoritmickou bez konkrétního jazyka a výuku programování ve Visual Basic. V algoritmické části, která je použitelná i pro jiné jazyky se žáci zaměřují na obecné znalosti z programování a algoritmizace bez zaměření na programovací jazyk Visual Basic. Je zde probírána problematika algoritmů, částí programů, deklarativním částem programu, příkazovým částem programu, funkcím, procedurám a jak se program vykonává. Také je zde uváděno, co je proměnná, jaké operace se s ní dají provádět a co je to příkaz a jak se s ním pracuje. V části programování ve Visual Basic, která již vychází z konkrétního programovacího jazyka je zaměřena na práci v aplikačním prostředí, formuláře, objekty, handlery, proměnné procedury a debuggery. Část programovací je taktéž zaměřena na tvorbu aplikací, navrhování formulářů, vhodný výběr událostí, práce se soubory, práce s databázemi a OLE technologie.

Samotná výuka v této koncepci je rozdělena do dvanácti tematických celků, kde některé jsou zaměřeny přímo na prostředí a specifika jazyka Visual Basic a ostatní na programování taktéž v tomto jazyce, ale s prvky, které obsahují i ostatní programovací jazyky. Jednotlivé celky zde nebudou detailně popsány, ale bude zde popsána opět jen posloupnost, ve které jsou žáci učeni.

V prvním celku se žáci učí pracovat s uživatelským rozhraním Visual Basic. Ve druhém se učí pracovat s ovládacími prvky prostředí a jejich základním nastavením. Je zde také uveden pojem objekt a jeho základní vlastnosti. Ve třetím celku přichází na řadu proměnné a podmínky. Žáci se učí, co jsou datové typy, jak se deklarují proměnné a konstanty. Po proměnných přichází na řadu větvení programu a různé možnosti zápisu. Čtvrtý celek je věnován práci s událostmi, jaké jsou jejich typy a jak se obsluhují. Pátý celek je věnován práci se soubory na disku počítače, jak se k nim přistupuje a jak lze procházet stromovou strukturu na disku. Šestý celek je věnován práci s menu, které se učí vytvářet. Další částí tohoto celku je práce s generovanou grafikou. V sedmém celku se pracuje s textovými soubory, kde se žáci učí přistupovat k jejich obsahu a je zde i zmíněna práce s chybami. Žáci se učí ladit program pomocí ladících nástrojů a učí se vyhledávat chyby. Problematika řetězců a cyklů je probírána v osmém celku. V části řetězců je problematika zaměřena na jejich zpracování a následné práce s nimi je zde i probíráno pole proměnných. V cyklech je problematika věnována různým druhům cyklů a jejich vhodnému nasazení v programech. Devátý celek je věnován práci s časem a datem. Je zde

prezentován základní přístup pro práci s datem a časem a zároveň funkce, které s těmito hodnotami umějí pracovat. Část tohoto celku je věnována problematice časovaní a jeho využívání v programu včetně nastavení. Celý desátý celek je věnován pokročilému využívání všech druhů cyklů a jednorozměrnému poli dynamických proměnných. Předposlední jedenáctý celek je věnován práci s dialogy a seznamy. Je zde ukázána pokročilá práce s dialogovými okny a využití pokročilých seznamů a jejich nastavování. Poslední dvanáctý celek je věnován základní práci s databázemi. Věnuje se jejich vytváření a práci s nimi pomocí Visual Basic. Problematika je zaměřena na připojení databáze k vytvořenému projektu a její doplňování, rozšiřování a zpracovávání pomocí příkazů.[45]

Druhou částí celé koncepce, která je především určena pro žáky, je její webová podpora. Ta je rozčleněna tak, jak metodická příručka popisovala taktéž do dvanácti celků, které jsou totožné s výše uvedenými. Jednotlivé celky jsou členěny do kapitol a podkapitol, které se zabývají rozpracováním učiva konkrétního celku. Rozpracované učivo bývá doplněno vhodnou ukázkou, která má grafickou nebo textovou podobu hodící se k probírané problematice. Jednotlivé kapitoly a podkapitoly jsou koncipovány ne jako vysvětlující, ale spíše demonstrující a sledující výklad učitele. Zpětně si v nich žáci mohou dohledat informace, o kterých se učili. Součástí elektronické podpory jsou cvičné příklady, které si žáci během hodin či doma mohou vyzkoušet vypracovat. Součástí je také metodický list, ve kterém je zobrazen konkrétní grafický výsledek příkladů včetně jeho možného řešení pomocí jazyka Visual Basic. Na konci probíraného celku se nachází zadání samostatné práce, které obsahuje vzorové zadání, vzorové zadání s chybami a samostatné zadání.[46]

3.3.6.2. Elektronická podpora programování

Poslední, často se objevující koncepcí v oblasti programování, jsou různě propracované webové stránky, které umožňují studium konkrétního jazyka. Mezi jazyky s největší podporou těchto technologií patří jazyky a nástroje spjaté s vývojem a programováním webových aplikací. Stránek zabývajících se touto problematikou je velká řada a dají se nalézt jak v českém, tak i v anglickém jazyce. Bezespornou výhodou při prezentaci

[45] KLEMENT, KLEMENT, LAVRINČÍK. *Metody realizace a hodnocení výuky základů programování. Olomouc*: Dostál Jiří 2012. ISBN 978-80-87658-01-7

[46] Klement, LAVRINČÍK. *Programování do škol.* [online]. PROŠ [cit. 2014-03-23] Dostupný z WWW: < http://www.pros.upol.cz/files/others/ucebnice-online-v3/index.htm>.

výukového materiálu je zároveň možnost konkrétních ukázek bez nutnosti instalace speciálního softwaru na straně klienta, což zvyšuje potenciál těchto koncepcí.

Webových stránek zabývajících se programováním je velká řada. V případě požadavku ozkoušení konkrétního programu přímo na webových stránkách se tato řada zužuje, a to díky složitosti emulátorů programovacích jazyků přímo na stránkách. Výhodně z těchto požadavků právě vychází webové technologie, které tyto emulátory nepotřebují. Dále zde bude popsána konkrétní koncepce, která umožňuje on-line odzkoušení kódu v PHP jazyce.

Koncepce je postavena jako výukový materiál především pro samostudium, bez výkladu učitele. Neobsahuje žádné zdlouhavé pasáže vysvětlující problematiku. Jedná se o krátké vysvětlení problému a konkrétní ukázku kódu, kterou si uživatel může dle vlastního uvážení pozměnit a vyzkoušet její funkčnost.

Celá koncepce je rozdělena do osmi částí, které se zabývají různou problematikou. Pro účely programování na základní škole, by byla stěžejní první část, která se zabývá základními principy programování v PHP včetně ukázek. Druhá část se zabývá formuláři, které by se při výuce webových aplikací na ZŠ daly také použít. Třetí část patří pokročilým technikám v PHP, čtvrtá databázím, pátá XML, šestá PHP a AJAX, sedmá konkrétním příkladům a osmá referenční příručce s jednotlivými příkazy a funkcemi.

Pro ilustraci možného postupu výuky programování na ZŠ zde bude uveden postup, který je v první části této koncepce. Celá koncepce začíná, tak jako některé předchozí výpisem textu typu „Hello World“. Pokračuje základní seznámení s problematikou PHP jazyka a jeho zakomponováním do struktury serverů a webových stránek, jak ho nainstalovat a co je k němu potřeba. Následuje základní vysvětlení syntaxe v PHP, a jak lze kombinovat výpis textu a různých dalších hodnot. Proměnné jsou vysvětlovány hned po problematice výpisu textu v kontextu globálních a lokálních proměnných a jejich využití. Pokud uživatel zvládá proměnné je přistoupeno k jejich výpisu a výpisu proměnných spolu s různými texty. Následuje další rozšíření proměnných a to uvedením jejich datových typů, které mohou používat. Jsou zde uvedeny typy string, integer, plovoucí desetinná čárka, boolean, pole a objekty. U typu string jsou uvedeny funkce, právě pro práci s tímto datovým typem. Následují konstanty a jejich správný zápis. Rozsáhlá kapitola je věnována operátorům a ukázkám jejich použití. Jako první jsou uvedeny aritmetické operátory, následují přiřazovací operátory, řetězcové operátory, porovnávací, logické a operátory pro práci s polem. Na kapitolu po operátorech navazuje větvení programu podmínkami včetně

několikanásobné podmínky. Po větvení programu se kapitola věnuje cyklům s podmínkou na začátku, podmínkou na konci a „for“ cyklus. Po zvládnutí cyklů studující přechází na funkce, jejich zápis, způsoby předávání a vracení hodnot. Poslední tři kapitoly jsou věnovány problematice polí, třídění dat v polích pomocí funkcí a superglobálním proměnným.[47]

3.3.7. Zhodnocení jednotlivých koncepcí

Společným znakem všech koncepcí je podobná posloupnost výuky základních programovacích částí, které jsou ve všech programovacích jazycích. Jedná se především o výpis textu nebo hodnot, proměnné, cykly, funkce či procedury, větvení programu, události. Každá koncepce k této problematice přistupuje hlavně podle připravených příkladů, které se postupně rozvijí a využívají již vysvětlenou problematiku a dále na těchto znalostech staví.

V knihách, které slouží pro výuku a ne jako referenční příručky bývá problematika dosti obšírně vysvětlována. Takto vysvětlená problematika se spíše hodí pro lidi s vyšší vnitřní motivací a zájmem o problematiku. Pro žáky na základní škole by nemusela být úplně pochopitelná a hlavně díky délce výkladu nezáživná. V tomto případě je spíše vhodná transformace obsahu učitelem do podoby vhodné pro konkrétní žáky. Vzhledem k prezentaci konkrétních příkladů programování, které jsou uvedeny v knize bez interaktivního propojení, je tento způsob pro žáky taktéž méně atraktivní. Přiblížení žákům by bylo možné dosáhnout tím, že by se jednalo přímo o učebnici, pro konkrétní věkovou kategorii. Někteří autoři se snaží tyto knihy uvést na trh, ale je jich málo a týkají se většinou propedeutických programovacích jazyků.[48]

Koncepcí, která odstraňuje výše zmíněné nedostatky knižní podoby a je zaměřená přímo na konkrétní skupinu žáků je klasická koncepce s příklady. Učivo je zde většinou redukováno do přijatelného množství a didakticky zpracováváno učitelem. Pak takovýto obsah slouží jako studijní podpora a nikoliv jako materiály pro samostudium. Jako nevýhoda se jeví opět pouze podoba, která je papírová nebo elektronická a neobsahuje žádné přizpůsobení příkladů pro moderní technologie.

---

[47] WS3SCHOOLS. *PHP Tutorial* [online]. [cit. 2014-03-24] Dostupný z WWW: <http://www.w3schools.com/php/default.asp>.

[48] BLAHO, KALAŠ. *Imagine Logo – programování pro děti*. Brno: Computer Press 2006, ISBN 80-251-1015

Nedostatek spočívající v nedostatečné podpoře příkladů a podání celého materiálu v elektronické podobě odstraňují koncepce metodických příruček s elektronickou podporou. Ty se vhledem k vývoji současných technologií směrem k digitálnímu online prostředí žákům začínají přibližovat a vytvářejí tak příjemnější a modernější prostředí včetně podpory pro výuku programování. Nevýhodou je opět malá dostupnost v českém jazyce pro konkrétní skupinu žáků. Některé z nich jsou dostupné z projektů, které byly ozkoušeny na Gymnáziích a pro běžnou výuku na ZŠ nejsou vhodné. Další nevýhodou v době růstu různorodosti operačních systémů je nutnost instalace programovacího jazyka i na domácí zařízení. Ne vždy lze konkrétní jazyk sehnat na všechny operační systémy. Pokud žáci nemohou v domácím prostředí procvičovat či se připravovat, jedná se o ztrátu výhody elektronické podpory.

Jako výhodnou koncepci, která odstraňuje nedostatky nutnosti instalace programovacího jazyka, se jeví elektronická podpora programování, která může být navržena jako různé emulátory programování v online prostředí nebo využívá webové technologie online. První zmíněná varianta online emulátorů programovacích prostředí není vždy spolehlivá z hlediska použitelnosti a jedná se o prostředí sloužící k vyzkoušení konkrétního jazyka a konkrétních programů. Většina online emulátorů neobsahuje ani výkladovou část, která by danou problematiku vysvětlovala. Druhá varianta webových technologií online je z tohoto pohledu propracovanější a také se dá nalézt větší množství takto propracovaných stránek. Výhodou je výkladová posloupnost, která bývá strohá, tudíž se dá použít jako podpůrný prostředek pro výuku programování ve zvoleném prostředí. Nevýhodou je spíše zaměření na více motivované jedince, kteří mají zájem celou problematiku prostudovat a pochopit. Pro žáky základní školy by se musely vybírat jen určité pasáže, které se hodí pro danou skupinu žáků. Jako další nevýhodu lze spatřovat, že příklady jsou v koncepci již vypracované a je na uživateli, aby si je vyzkoušel a upravil a ne aby vymýšlel svůj algoritmus. Posledním zásadním problém je většinová dostupnost těchto stránek pouze v anglickém jazyce, což pro žáky základní školy také není vhodné.

Každá zde uvedená koncepce má své výhody a nevýhody. Každá koncepce je použitelná a efektivní pro určitou skupinu lidí a její paušalizované využití není tudíž vhodné. Žádnou z koncepcí nelze zamítnout, protože si svou skupinu lidí, kteří ji využijí, nalezne. Zároveň lze konstatovat, že některé z nich jsou více či méně vhodnější pro výuku programování žáků na základní škole.

4. Návrh vlastní koncepce

Na základě ústního šetření mezi učiteli informatiky, kteří vyučují nějaký programovací jazyk ve svém předmětu, lze dospět k názoru, že by při výuce uvítali možnost, která by umožňovala problematiku programování žákům více přiblížit. Jako výhodu by viděli více motivované žáky, kteří by projevovali větší zájem o výuku a kteří by dosahovali lepších výsledků. Zároveň by viděli pozitivně řešení, které by mělo online podporu poplatnou dnešní době. Z těchto požadavků a z analýzy dostupných koncepcí lze stanovit obecné požadavky na navrhovanou koncepci.

Vzhledem k posloupnosti výuky (výpis textu nebo hodnot, proměnné, cykly, funkce či procedury, větvení programu, události), která se objevuje u všech koncepcí, lze dospět k názoru, že se jedná o přijatelný postup, který je uplatňován v široké míře a lze ho i u navrhované koncepce využít s přizpůsobením konkrétnímu jazyku a úrovni žáků. Navíc se jedná o obecná témata, které lze využít v jakémkoliv programovacím jazyce. Koncepce by měla mít online podporu pro žáky, která by sloužila k podpoře výuky, upevňování znalostí a opakování ve škole či v domácím prostředí. Velkou výhodou by byla možnost využití koncepce na jakémkoliv zařízení bez ohledu na operační systém a nainstalované aplikace. Výuka by měla obsahovat příklady, které jsou pochopitelné a přiměřené úrovni žáků na základní škole. Příklady by měli zvládnout jak slabší žáci, tak i nadanější žáci, kteří by měli mít možnost vypracovávat složitější příklady.

4.1. Výběr vhodného jazyka

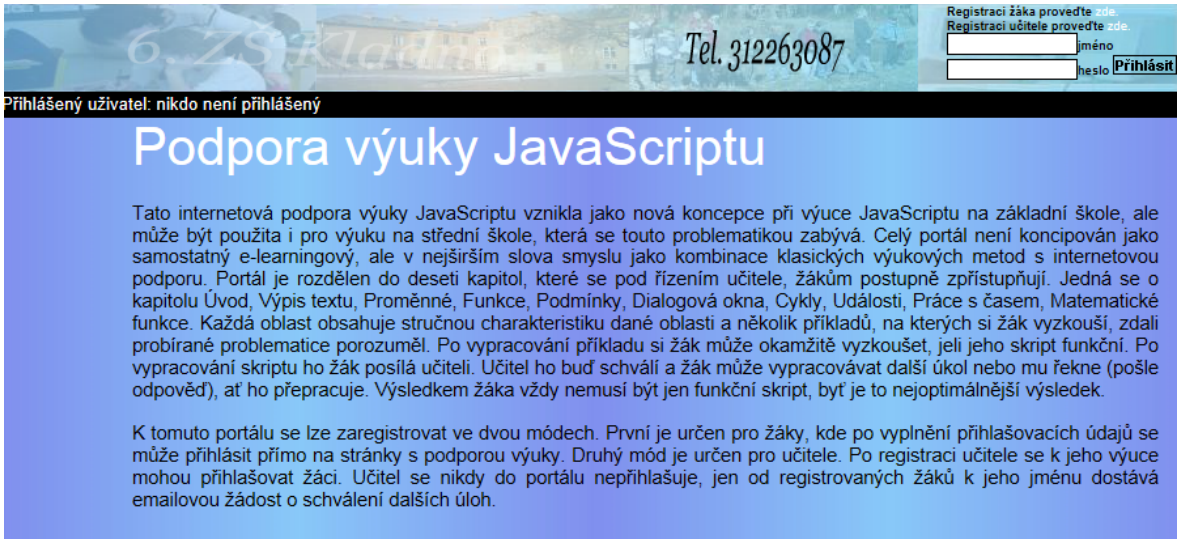
Ve škole, ve které bude probíhat zkušební výuka dle nové koncepce, jsou žáci od páté třídy rozděleny na žáky zaměřené na informatiku a na ekologii. V informatické větvi se žáci setkávají s informačními technologiemi od čtvrtého do devátého ročníku. Školní vzdělávací program v tomto ohledu obsahově naplňuje požadavky rámcového vzdělávacího programu a navíc ho doplňuje o témata, která jsou v dnešní době potřebná. Mezi tato témata kromě jiných patří algoritmizace a programování, jež je postupně naplňováno od sedmého ročníku základní školy. V sedmém ročníku mají žáci naplánován značkovací jazyk HTML, ve kterém se učí vytvářet jednoduché stránky, učí se analyzovat předložený kód a upravovat ho, aby vznikl požadovaný výsledek. V osmém ročníku na téma HTML navazují CSS, které umožní žákům problematiku formátování stránek vidět z jiného pohledu a rozšíří si tak možnosti formátování stránek a obsahů modernějším

způsobem. V devátém ročníku si žáci rozšiřují znalosti o webových stránkách pomocí skriptovacího jazyka JavaScript a na koci ročníku úvodem do PHP.

Po analýze vzdělávacího programu a požadavků vyplývajících z vlastního návrhu koncepce se jako vhodný jazyk splňující požadovaná kritéria jeví JavaScript. Jedná se sice o skriptovací jazyk, ale mezi jeho výhody patří objektové pojetí, možnost využití nejčastější posloupnosti výuky a především jeho možné odzkoušení na většině internetových prohlížečů bez potřeby konkrétního operačního systému. Tím se otevírá prostor pro jeho využití na tabletech a chytrých telefonech, které jsou mezi žáky oblíbené.

4.2. Technický popis portálu

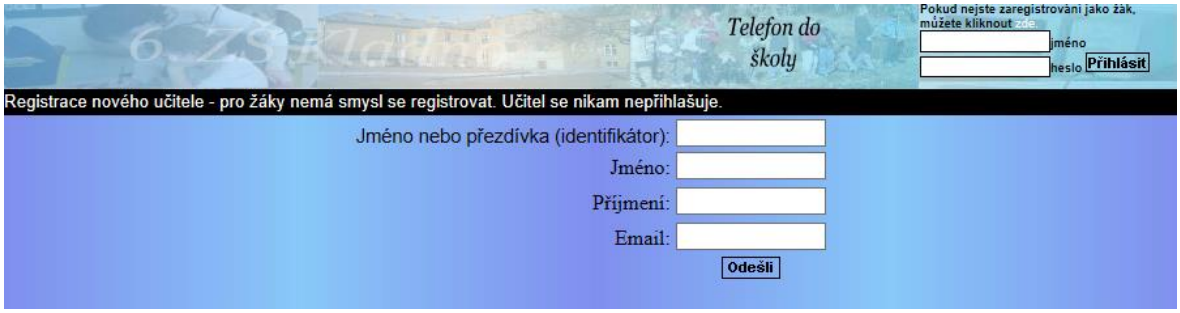
Pro zvolenou koncepci byl navržen technický portál, který splňuje požadavky vycházející z jednotlivých analýz. Celý portál je postaven na webových technologiích a pro komunikaci s uživatelem používá webové stránky. V současné době portál běží na webovém serveru IIS, který umožňuje skriptování na jeho straně v jazyce PHP a zároveň využívá pro ukládání dat databáze. Celý portál je složen ze čtyř modulů, které slouží k jeho používání. Jedná se o hlavní modul, registrační modul pro žáky, registrační modul pro učitele a postupový modul. Všechny tyto moduly jsou uloženy jako skripty jazyka PHP a ke své činnosti využívají data ze souběžně běžícího serveru MySql. Výsledkem skriptů jsou pak webové stránky obsahující HTML, CSS a JavaScript.



Obrázek 2: Úvodní stránka portálu

4.2.1. Registrační modul učitele

V případě prázdné databáze je nutné před jakýmkoliv používáním portálu využít první modul, pomocí kterého se registruje učitel. Modul obsahuje celkem čtyři povinná pole, která musí učitel vyplnit. Jedná se o jméno nebo přezdívku, která slouží jako identifikátor učitele. V současné době je položka používána pouze jako jedinečný identifikátor, ale v případě doplnění dalších modulů by mohla sloužit jako přihlašovací jméno. Druhou položkou je jméno učitele, třetí příjmení učitele a poslední čtvrtou položkou je email učitele. Všechna data jsou ukládána do samostatné tabulky databáze a je zjišťována duplicita jedinečného identifikátoru. Tato data se využívají při registraci nového žáka v dalším uvedeném modulu.



Obrázek 3: Registrační modul učitele

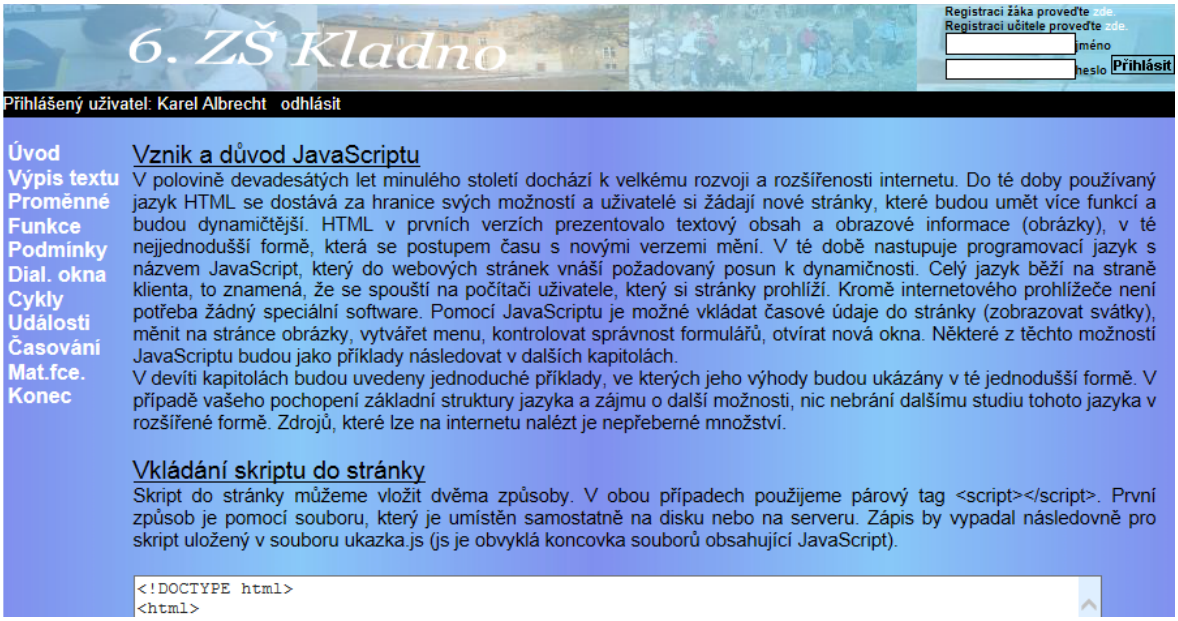
4.2.2. Registrační modul žáka

Registrační modul žáka slouží pro samotnou registraci žáka do celého systému. V tomto modulu je celkem šest položek, které je potřeba vyplnit. Všechny tyto údaje jsou zapsány do tabulky žáků. Jako jedinečný identifikátor je zde použito přihlašovací jméno, pomocí kterého se i žáci do celého systému logují. Druhou položkou je heslo, sloužící k ověření přihlašovaného. Třetí položka je výběrová a je propojena s tabulkou učitelů, ze které čerpá informace o zaregistrovaných učitelích. Čtvrtá položka je žákovo jméno, pátá žákovo příjmení a poslední žákův email. Všechny zde popsané položky jsou dále portálem používány pro vzájemnou komunikaci. Pro uložení informací o uživateli je použita databázová tabulka žáků obsahující výše uvedené položky. Dalším záznamem v tabulce žáků, který je potřeba pro správnou funkci celého portálu, je číslo poslední úlohy, které je důležité i pro další moduly.

Obrázek 4: Registrační modul žáka

4.2.3. Hlavní modul

Nejdůležitější částí celého portálu je hlavní modul, který umožňuje žákům přistupovat k výukovému obsahu. Mezi nejdůležitějšími úkoly jsou autentifikace uživatele podle tabulky žáků, zpřístupnění výukového obsahu podle čísla úlohy v tabulce žáků a generování postupových emailů využívající včetně tabulky žáků i tabulku učitelů. Celý hlavní modul v sobě obsahuje textovou část výuky, všechny příklady a všechny správné možnosti výsledků a stává se tak sjednocujícím pro všechny moduly. Modul generuje postupové emaily, které jsou zasílány konkrétnímu učiteli.

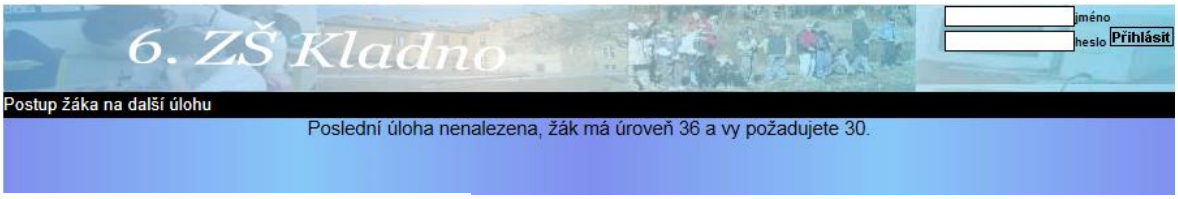


Obrázek 5: Hlavní modul

4.2.4. Postupový modul

Postupový modul je vázán pouze na postupové emaily z hlavního modulu a sám o sobě by jako webová stránka bez vstupních údajů byl neaktivní. Tento modul přijímá metodou

GET potřebné údaje o žácích, které zasílá konkrétní učitel žáka. Modul zkontroluje aktuální úlohu žáka a zhodnotí, zdali je možné zapsat novou úlohu. Modul zároveň přepisuje hodnotu čísla poslední úlohy v tabulce žáků.



Obrázek 6: Postupový modul

4.3. Funkční popis portálu

Jak bylo popsáno v předchozí kapitole technického popisu portálu, celý se skládá ze čtyř modulů, které se díky hypertextovým odkazům prolínají, a není mezi nimi vidět rozdíl. Z tohoto důvodu zde bude celá koncepce popsána jako jeden funkční celek. Prvním krokem při práci s portálem, který musí učitel učinit, je jeho registrace do portálu. Ta slouží k uchování informací o jeho osobě a k uchování emailu, na který budou následně zasílány postupové emaily. Po registraci se učitel již do portálu nikdy nepřihlašuje, z tohoto důvodu není potřeba vyplňovat žádné heslo, ale zároveň je zde ponechána možnost v budoucnu rozšiřovat moduly dalším jejich doprogramováním.

Druhým krokem po registraci učitele je registrace žáka. Každý žák vyplní své přihlašovací jméno, skutečné jméno a heslo. Následně si vybere učitele, se kterým bude výuka probíhat a který ho bude celým výukovým procesem provázet. Žák také musí vyplnit svůj funkční email, který slouží ke komunikaci s ním, pokud vypracovává úlohu mimo školu. Zároveň na uvedený email dostává informace o tom, že mu byl umožněn postup k další úloze.

Po registraci žáka může také začít proces výuky. Podpůrná a procvičovací část je rozdělena do deseti kapitol, které obsahují 35 povinných úloh. Každá kapitola je zaměřena na specifickou problematiku programování.

Po zvládnutí určitého úseku, který je ve většině případů ověřen příkladem, portál zasílá výsledek v podobě emailu na učitelem uvedenou adresu. Email obsahuje informace o žákovi, který žádá o postup k další úloze, obsahuje jeho emailovou adresu, na kterou může učitel odpovědět a hlavně obsahuje výsledek žákovské práce. Pokud se učitel rozhodne, že žák může postoupit k další úloze, zvolí tuto možnost v emailu, který dostal. Automaticky je generován email, který upozorní žáka, že má možnost pokračovat v další úloze.

Vyzvednutí emailu žákem není podmínkou, pouze upozorněním, že může pokračovat. V portálu stačí aktualizovat stránku nebo v navigačním panelu zvolit možnost „Email byl odeslán. Po schválení úlohy klikni zde.“ Celý popsaný proces se opakuje do té doby, dokud žák neprojde celou výukou.

4.4. Obsahově didaktický popis portálu

Obsah portálu vychází z předešlé analýzy, především z trendů ve výuce programování. Jeho zaměření je sice na JavaScript, ale byly vybrány především ty kapitoly, které jsou společné pro všechny programovací jazyky, aby žáci mohli kdykoliv v budoucnu přejít na jiný programovací jazyk a zároveň znali základní podstatu programování. Jelikož celá koncepce bude ověřována na běžné základní škole, je zde kladen důraz, aby co největší množství žáků bylo schopno problematiku pochopit.

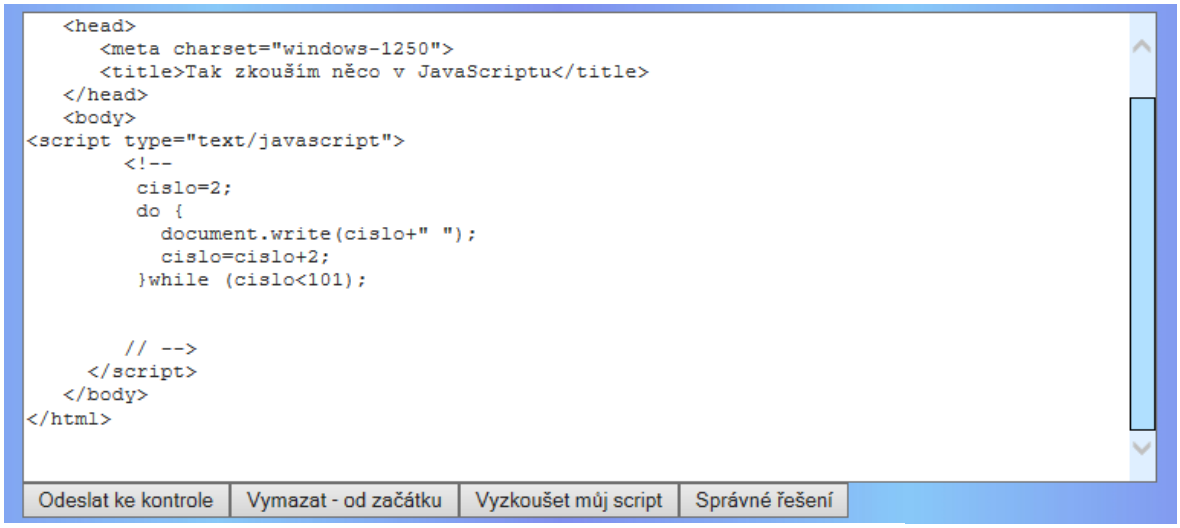
Portál z didaktického hlediska obsahuje jednak podpůrnou část, která je především určena pro žáky, aby například mohli sledovat výkladovou část učitele, dále pak aby si mohli již probrané učivo kdykoliv upevnit a zopakovat. V krajním případě je určena i pro žáky, kteří chyběli nebo chybí a danou problematiku studují sami. Druhou možností využití podpůrné části je využití pro učitele, kteří mohou zjistit, co je obsahem daného tématu a kam směřovat své působení.

Druhá didaktická část portálu obsahuje konkrétní úlohy, které slouží k procvičování a upevňování znalostí probíraného učiva. Celý portál obsahuje celkem třicet pět úloh, které jsou začleněny do devíti probíraných kapitol. Kromě těchto úloh obsahuje ještě devět dobrovolných úloh, které jsou svým obsahem složitější a slouží pro žáky s hlubším zájmem o probírané učivo a pro nadané žáky. Na jejich vyřešení je někdy potřeba dostudovat malou část programovacího jazyka z jiných zdrojů.

Zadání a řešení úloh je realizováno pomocí internetového prohlížeče a formulářových polí. Toto řešení se zdá být optimální z hlediska použití na různých platformách a operačních systémech. Pod formulářovým polem se nachází tlačítko „Odeslat ke kontrole“, které umožňuje poslední úlohu odeslat učiteli na zhodnocení. Po odeslání se v navigačním panelu zobrazí „Email byl odeslán. Po schválení úlohy klikni zde.“ Pak už jen uživatel čeká na reakci učitele, která může být slovní, kdy učitel sdělí žákovi, že má úlohu schválenou a může pokračovat dál. Další možností je, že učitel řekne připomínky, popřípadě nechá žáka úlohu přepracovat. Pokud žák z nějakého důvodu není přítomný ve

škole nebo si dodělává úlohy například doma, může mu učitel odpovědět pomocí emailu. Každopádně při schválení úlohy dochází k automatickému generování emailu o postupu, který je doručen žákovi.

Druhým tlačítkem pod formulářem je tlačítko „Vymazat – od začátku“, které celý formulář vymaže, jen v něm nechá základní části programu, aby je uživatel nemusel pokaždé vytvářet od začátku. Třetí tlačítko se nazývá „Vyzkoušet můj script“ a slouží právě k tolik chtěnému vyzkoušení naprogramovaného skriptu v podstatě na jakékoliv platformě. Poslední tlačítko „Správné řešení“ je přístupné pouze u schválených úloh a slouží žákům, aby se mohli kdykoliv v budoucnu vrátit zpět k jakékoliv úloze a podívat se na možný způsob jejího řešení. Vzhledem ke zkušenostem, že žáci správné řešení, které znají před samotným řešením úlohy, často opíší a nesnaží se vymyslet vlastní řešení, je toto tlačítko v programu přístupné až po splnění a odsouhlasení úlohy učitelem.

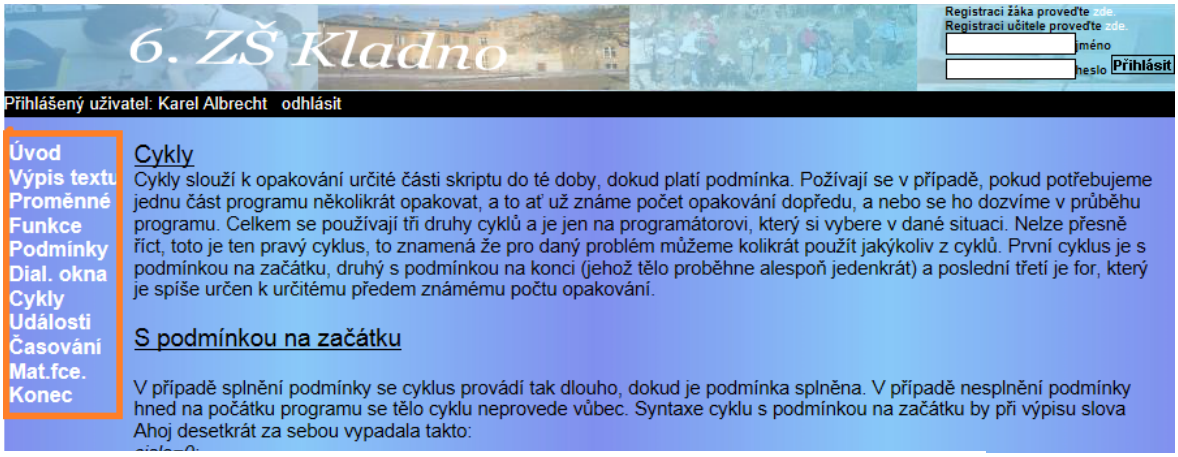


Obrázek 7: Formulář pro vypracovávání úloh, tlačítka

Výuka v jazyce JavaScript je koncipována pro úplné začátečníky, není tedy potřeba jakýchkoliv předchozích znalostí z oblasti programování. Zároveň je nutno dodat, že se jedná o webový skriptovací jazyk, tudíž jsou pro správné a hlavně smysluplné plnění úloh nutné základní znalosti značkovacího jazyka HTML nebo jejich základní vysvětlení učitelem, před samotným zadáním úlohy.

4.4.1. Obsah podpůrné části

Podpůrná část je rozdělena celkem do deseti kapitol, které na sebe navazují. Znalosti z jednotlivých kapitol jsou využívány v kapitolách dalších.



Obrázek 8: Kapitoly, které se postupně zpřístupňují

První kapitola „Úvod“ je určena pro základní seznámení s jazykem JavaScript a vysvětlení jeho postavení při tvorbě webových stránek. Z didaktického hlediska je přínosnější jeho zasazení do konkrétního prostředí, než ho probírat izolovaně od jeho primárního určení. Je zde také popsáno, že se jedná o technologii běžící na straně klienta a jaké to má výhody. Druhá část úvodní kapitoly je už konkrétně věnována vkládání skriptu do webových stránek. Jsou zde popsány dva způsoby, jak vložit skript do webových stránek a to buď jako externí soubor, na který směřuje uvedená cesta nebo přímo psaný skript v rámci konkrétní stránky.

Druhá kapitola „Výpis textu“ je věnována právě této problematice, která se zdá být na první pohled jednoduchá, ale její správné zvládnutí je důležité v dalších fázích programování a výpisů výsledků. Je zde uvedena základní syntaxe pro výpis textu a její možné varianty dle požadavků programátora. Důležitou částí je spojení vypisovaných znaků, které je popsáno buď jako spojování dlouhých vět, anebo spojování částí, kde dochází ke kombinaci textu a výpočtů.

Třetí kapitola „Proměnné“ je věnována problematice proměnných a vůbec uvedením toho, co to proměnná je, co si pod tímto pojmem představit. Jazyk jako takový striktní deklaraci proměnných nevyžaduje a z tohoto důvodu nejsou detailně uváděny jednotlivé datové typy. Je zde uvedeno, že údaje, které jsou uložené v paměti, mohou obsahovat číselnou hodnotu, se kterou je dále možno provádět základní matematické operace. Druhou možností, se kterou se žáci budou v dalších kapitolách setkávat, je textová hodnota proměnné. Je zde uvedeno, co je to textová hodnota a jaké údaje může obsahovat. Třetím typem proměnné je logická proměnná, která nabývá hodnot true nebo false. Poslední typ proměnné, který je zde uveden, je pole včetně ukázky jeho naplnění. Po naplnění je ukázán i zpětně přístup

k jeho jednotlivým položkám. Důležitou částí, která je zde zmíněna, je pojmenovávání proměnných, aby nedocházelo ke konfliktům při jejich vytváření a v neposlední řadě syntaxe pro naplnění konkrétní proměnné hodnotami.

Čtvrtá kapitola se zabývá problematikou funkcí. Funkce jsou zde rozděleny do dvou skupin a to na funkce s argumentem a funkce bez argumentu. Je zde uvedeno, co je to vlastně funkce, k čemu slouží a kdy je vhodné ji využít. U funkce bez argumentu je ukázán její obecný zápis. U funkcí s argumentem je uvedeno jak argumenty předávat do funkce a kolik jich může být, že nemusí být pouze jeden.

Pátá kapitola s názvem „Podmínky“ je řeší větvení programů dle požadovaných kritérií. Je zde popsáno k čemu podmínky slouží a kde se dají využít. Zároveň je zde použito grafické znázornění větvení programů pro lepší představu žáků, co jsou to podmínky a k čemu slouží. Je zde uvedeno šest základních operátorů včetně konkrétního příkladu, aby si žáci mohli vytvořit představu o konkrétním výsledku, a zhodnotit zdali je podmínka splněna či ne. Posledním tématem čtvrté kapitoly jsou logické operátory a to vysvětlení logického součtu a součinu včetně rozšíření ukázkového příkladu o další ukázku s použitím těchto operátorů.

Šestá kapitola představuje dialogová okna, která jsou specifická pro JavaScript. Dialogová okna jsou zařazena z důvodu, aby žáci mohli zadávat hodnoty do jejich programu a dále s nimi pracovat. Druhým důvodem je používání tohoto způsobu zadávání hodnot i některými webovými stránkami, tudíž se jedná o motivační prvek. Je zde uvedeno výstražné okno a jeho možný zápis. Dalším oknem je potvrzovací okno, které vrací dvě hodnoty. Posledním dialogovým oknem s největším potenciálem využití při výuce programování je vstupní dialogové okno, do kterého lze zadávat libovolná data dále ke zpracování.

Sedmá kapitola „Cykly“ je věnována opakování určité části programu. Jsou zde uvedeny tři základní druhy cyklů, se kterými se žáci při programování mohou setkávat. První zde uvedený cyklus je s podmínkou na začátku a ukázka jeho syntaxe. Druhým cyklem je cyklus s podmínkou na konci s opět uvedenou syntaxí a uvedení rozdílu mezi těmito cykly. Posledním druhem cyklu je cyklus „for“, a možnost jeho využití.

Osmá kapitola je věnována událostem. Jedná se o malý náhled do problematiky objektového programování a jeho možného využití při tvorbě webových stránek. Je zde uvedeno několik základních událostí, které jsou jednoduše pochopitelné a využitelné při

tvorbě stránek. Jedná se o události „onClick, onDblClick, onLoad, onUnLoad, onMouseOver, onMouseOut, onResize, onKeyDown“. Důležitá je ukázka možného zápisu událostí do struktury samotné stránky, kde dochází k propojení jazyka HTML a JavaScript, což přináší tížený výsledek.

Devátá kapitola poukazuje na standartní problematiku programovacích jazyků a to časování. Je tu uveden způsob, jak se dostat k systémovému času a jak se s ním dá dále zacházet. Konkrétní ukázkou je demonstrováno, jakým způsobem lze pomocí časových funkcí, mezi které patří především: getFullYear(), getMonth(), getDate(), getDay(), getHours(), getMinutes() a getSecons(), získat potřebný údaj ze systémového času. Druhá část této kapitoly je zaměřena na časování, které slouží k tomu, aby se v určitém časovém intervalu provedl například nějaký příkaz či spustila funkce.

Poslední desátá kapitola je věnována matematickým funkcím, které se používají opět ve většině programovacích jazyků. Kapitola byla navržena dle analýzy školního vzdělávacího programu konkrétní ZŠ, tudíž může dojít k situaci, že obsah zde prezentovaný nemusí odpovídat učivu v matematice. Z tohoto důvodu může být aplikace některých částí matematických znalostí v programování nad rámec znalostí v matematice. V kapitole jsou uvedeny základní matematické funkce a správný zápis syntaxe, aby bylo možno s nimi pracovat. Mezi ukázané funkce patří: sin(), cos(), tan(), round(), abs(), random(), sqrt(), a konstanta PI.

4.4.2. Popis příkladů

Část portálu věnovaná úlohám obsahuje celkem třicet pět povinných a devět volitelných úloh.

První kapitola neobsahuje žádnou úlohu, je v ní nastíněn způsob postupu do další kapitoly v podstatě ukázka, jak se portál ovládá. Druhá kapitola již obsahuje čtyři povinné úlohy. V první žáci zkouší klasickou ukázku výpisu textu, jedná se vlastně o první metodu objektu document, kterou si vyzkouší a pomocí níž vypíšou text. Druhá úloha již obsahuje kombinaci výpisu textu a HTML značek a základnímu formátování. Je na ní doloženo, že je možné tento jazyk využívat i v rámci JavaScript. Třetí úloha v zadání obsahuje klasický výpis textu plus matematický výpočet, na kterém si žáci ozkouší, že se dá kombinovat text a matematické výpočty. Ve čtvrté úloze si žáci procvičí spojování jednotlivých částí výpisů textů, sloužící k úspoře délky kódu při programování. Dobrovolná úloha tohoto celku je

věnována většímu začlenění webových prvků do výpisu pomocí JavaScript. Žáci se pokouší vytvořit formátovaný text pomocí atributů tagů HTTML či pomocí CSS.

Třetí kapitola „Proměnné“ obsahuje dalších pět úloh. Pátá úloha je zaměřena na ukládání hodnot do proměnné. Žák má vytvořit celkem dvě proměnné *a*, *b*, do kterých ukládá číselné hodnoty a nakonec nechá vypsat jejich součin. Šestá úloha rozvíjí předcházející příklad tím, že je zde zavedena ještě jedna proměnná *vysledek*, do které je uležena matematický výsledek dělení *a, b* a opět je vše vypsáno. Sedmá úloha je věnována ukládání textů do proměnných a jejich možné spojování. Jsou zde opět použity tři proměnné *a, b, c*, kde do *a, b* je uložen text a v proměnné *c* je spojen. Osmá úloha je věnována proměnným typu pole, kde si žáci vyzkouší vytvořit pole obsahující měsíce roku a pomocí číselné hodnoty napsat o jaký měsíc se jedná, který předcházel a který bude následovat. Devátá úloha je opět zaměřena na pole, tentokrát na číselné údaje, které jsou v poli uloženy a se kterými žák provádí základní matematické operace. Dobrovolná úloha této kapitoly je věnována opět proměnné typu pole, kde do pole jsou zadávány matematické výsledky výpočtů s jinými proměnnými.

Čtvrtá kapitola „Funkce“ obsahuje další tři povinné úlohy, z nichž desátá v pořadí demonstruje výhody funkcí. Jedná se o vytvoření grafického oddělovače, který se dá použít mezi vypisovaným textem. Úloha je zaměřena na procvičování funkcí bez argumentu. Jedenáctá úloha již argumenty používá a jedná se o tvorbu grafického oddělovače, ve kterém je možno zvolit jeho barvu pomocí argumentu funkce. Poslední dvanáctá úloha je zaměřena na vytvoření obecné funkce, která umí sčítat libovolná tři čísla, která jsou předána v argumentech. Dobrovolná úloha je zaměřena na vracení hodnot přímo funkcí. Jde o funkci, která hodnoty přímo nevypisuje, ale jen vrací. V zadání je uveden směr, kterým se dá úloha řešit, ale je potřeba například přesnou syntaxi dohledat v literatuře.

Pátá kapitola „Podmínky“ obsahuje další tři úlohy. Třináctá je zaměřena na základní větvení programu dle hodnot proměnné. Při hodnotách v určitém intervalu dojde k vypsání určitého textu a při hodnotách z jiného intervalu k vypsání jiného textu. Čtrnáctá úloha je rozšířením úlohy předcházející o třetí větev, kdy jsou vytvořeny celkem tři intervaly, a každý vypisuje jiný text. Patnáctá úloha je zaměřená obdobným směrem jako předcházející, ale na základě vstupu, kterým je věk ve formě čísla, má žák vypsat celou větou, kolik je dotyčnému let ve správném gramatickém znění. Například: „Je ti 1 rok“

nebo „Je ti 5 let.“ Dobrovolná úloha je zaměřena matematicky, kde žák má vymyslet program, který bude zjišťovat, zdali zadaná proměnná spadá do zadaného intervalu či ne.
Šestá kapitola „Dialogová okna“ obsahuje další tři úlohy. Šestnáctá úloha je zaměřena na výstražné okno, kde si žák vyzkouší vytvořit dvě výstražná okna, která se budou postupně objevovat po sobě. Sedmnáctá úloha obsahuje potvrzovací okno, pomocí kterého se do proměnné uloží hodnota, zdali je splněno nebo ne. Dle výsledku v proměnné program rozhodne o dalším postupu, vypíše buď jeden, nebo druhý text. Osmnáctá úloha je zaměřena na vstupní okna. Žák má vytvořit program, který po něm chce zadat jeho věk a po jeho zadání dojde k rozvětvení do tří částí a k výpisu textu, do jaké věkové kategorie patří. Dobrovolná úloha je zaměřena na aplikaci Pythagorovy věty do programovacího prostředí. Žáci zadají délku přepony a odvěsny a program dopočítá druhou odvěsnu.
Sedmá kapitola „Cykly“ obsahuje další čtyři úlohy. V devatenácté úloze žáci programují cyklus s podmínkou na konci, který vypíše sudá čísla do sta. Dvacátá úloha je zaměřena na cyklus s podmínkou na začátku, který vypíše postupně čísla od 0 do -100. Dvacátá první úloha už demonstruje sílu cyklů v kombinaci s jazykem HTML. Úkolem žáků je vytvořit cyklus, který zobrazí ve své podstatě libovolný počet obrázků na webové stránce. Je zde demonstrován rozdíl v zápisu mezi klasickým způsobem pomocí HTML, kde je pro každý obrázek potřeba samostatný tag a cyklem v JavaScript, kde je pomocí několika řádků možné zobrazit libovolný počet obrázků. Dvacátá druhá úloha je kombinací dialogových oken a cyklu „for“. Žáci vytváří program, do jehož vstupu zadají první číslo intervalu a poslední číslo intervalu. Výsledkem je součet všech čísel obsažených v intervalu. Dobrovolná úlohy je cyklus v cyklu, který vypíše všechna čísla malé násobilky.
Osmá kapitola „Události“ obsahuje další tři úlohy. V dvacáté třetí úloze žáci vytváří program, který při dvojkliku na obrázek zobrazí dialogové okno s hláškou „Klikls dvakrát“. V další dvacáté čtvrté úloze žáci programují reakce na přejetí myší, která opět vyvolá dialogové okno. V dvacáté páté úloze jsou obsaženy části z předchozích úloh a jsou zde opět demonstrovány možnosti jazyka JavaScript. Žáci vytváří program, který po najetí myší na obrázek vyvolá událost, která ho změní na jiný. Po sjetí z obrázku dojde ke změně na obrázek původní. Dobrovolná úloha je zaměřena na události dvojkliku, které vyvolají zobrazení dalšího obrázku z připravené série. V podstatě žáci vytvoří mini prohlížeč obrázků.

Předposlední devátá kapitola „Časování“ obsahuje celkem dalších pět úloh. Dvacátá šestá úloha je zaměřena na výpis aktuálního času v zadání uvedeném formátu. Žáci pomocí časových funkcí postupně zjišťují hodinu, minutu a sekundu. Dvacátá sedmá úloha je kombinací proměnných typu pole a zjišťování aktuálního měsíce pomocí časových funkcí. Žáci zde vytváří program, který vypíše v českém jazyce název aktuálního měsíce. Dvacátá osmá úloha je zaměřená na časování. Žáci vytváří program, ve kterém je napsáno slovo a po zvoleném časovém intervalu dojde k napsání dalšího slova. Dvacátá devátá úloha je kombinací úloh pracujících s časem, kdy žák vytváří program, který vypíše aktuální datum v uvedeném formátu s použitím pole, ze kterého jsou načítány české dny z týdne a pole, které obsahuje české měsíce. Třicátá úloha je zaměřena na časování. Žáci vytváří program, který zobrazí obrázek a za pět sekund zobrazí obrázek jiný, po uplynutí opět pěti sekund, se zobrazí původní obrázek. Dobrovolná úloha vyžaduje od žáků malý náhled do literatury nebo přesné prostudování zadání úlohy. Zde žáci vytváří program, který bude na stránce zobrazovat přesný čas v základním formátu, ale bude docházet k jeho pravidelné aktualizaci každou sekundu.

Poslední desátá kapitola „Matematické funkce“ obsahuje dalších pět úloh. Třicátá první úloha je kombinací dialogového okna, pomocí kterého se zadává poloměr kruhu. Výsledkem programu je pak obvod a obsah kruhu vypočítaný pomocí matematické konstanty. Třicátá druhá úloha opět slouží k matematickým výpočtům, tentokrát k převodům ze stupňů na radiány. Do dialogového okna jsou zadány stupně a výsledkem programu je výpis radiánů. Ve třicáté třetí úloze žáci vytváří program, který vypočítává sinus a cosinus ze zadaných radiánů. Třicátá čtvrtá úloha je kombinací dvou předcházejících úloh a žáci zde mají za úkol vytvořit takový program, který ze stupňů vypočítá jejich hodnotu sinu a cosinu. Poslední třicátá pátá úloha je kombinací dialogových oken, cyklů a funkce pro generování náhodných čísel. Žáci vytváří program, kde do dialogového okna je zadán počet náhodně generovaných čísel a program je generuje a vypisuje. Poslední dobrovolná úloha obsahuje vygenerování padesáti šesti náhodných čísel a dle nich zobrazení obrázků. Při prvním pokusu zjistí, že se některé obrázky opakují a jejich úkolem je zkusit vymyslet program, který každý obrázek zobrazí pouze jednou.

4.5. Didaktický postup

Celá koncepce, jak vyplývá z předešlých kapitol, není určen jako e-learningový kurz nebo distanční studijní materiál. Jeho použití záleží na konkrétní situaci a konkrétním učiteli. Každopádně je doporučen postup, který buď v jedné či více hodinách bude obsahovat všechny fáze motivační, expoziční, fixační, diagnostickou a aplikační. Samozřejmě je na každém pedagogovi a jeho citu, jak v dané situaci bude postupovat.

Vhodnou organizační formou, co se týká časového hlediska, je dvouhodinová vyučovací jednotka, při které lze jednu kapitolu s žáky během této doby zvládnout. V případě kratší jednotky je potřeba počítat s minimálně dvěma vyučovacími hodinami na jednu kapitolu.

Doporučeným postupem v této koncepci je na začátku hodiny zařadit motivační část, ve které bude žákům nastíněno, co je v dané kapitole čeká a jaké praktické využití kapitola má, případně konkrétní ukázky některých výsledků. Po úvodní motivaci je možno přistoupit k expoziční části, která by se měla obsahově opírat o konkrétní podklady a příklady uvedené v dané kapitole. Rozsah tématu uvedený v portálu byl volen z osobních zkušeností s výukou programování na základní škole a měl by být zvládnutelný pro většinu žáků. Témata nepokrývají veškerou problematiku, jsou spíše brána jako úvod do konkrétní problematiky. Učitel může i v jednotlivých tématech vypustit z jeho pohledu složitější nebo nepotřebnou látku a následně jen přeskočit s žáky příklady, které se této problematiky týkaly. Expoziční část učitele by měla být čistě v jeho kompetenci a měl by ji vést tak, aby žáci danou problematiku pochopili. Podpůrné texty nejsou plnohodnotným materiálem nahrazující učitele, ale jen možností pro zopakování a upevnění si znalostí žáky. V expoziční části hodiny může učitel ukazovat žákům různé konkrétní příklady, které si žáci mohou přímo v portálu zkoušet vytvořit v jakékoliv úloze bez odeslání.

V další fázi výuky žáci přechází ke konkrétní tvorbě programů, pomocí připravených úloh. Každá úloha má písemné zadání plus u každé úlohy je uveden grafický výsledek, jak by měl vypadat výsledek programu. Grafický výsledek je důležitým doplněním slovního zadání a každý žák by si ho měl před tvorbou programu prohlédnout. Úlohy v širším pojetí plní zbývající tři funkce a to fixační, diagnostickou a aplikační. Každý žák má přístupné pouze úlohy, které mu byly učitelem schváleny. Respektive vždy se mu zobrazí další úloha, kterou má vypracovávat a poslat ke schválení učiteli. Pokud tedy žák vypracuje úlohu a pošle jí ke schválení svému učiteli, tak ten pak dostane výsledek úlohy na email a rozhodne, zdali úlohu žákovi schválí či ne. V případě schválení úlohy, má žák k dispozici

úlohu další a zároveň je mu zpřístupněno možné správné řešení předešlé úlohy, na které se může kdykoliv v budoucnu podívat. O výsledku schválení či neschválení může učitel žáka informovat ústně během vyučovacích hodin a může vyžadovat například doplnění či přepracování úlohy. Druhou možností je zaslání emailu, ve kterém popíše požadavky na žáka, tento způsob je vhodnější, pokud si žák dodělává doma chybějící úlohy, třeba z důvodu nemoci. Při schválení úlohy učitel nemusí žákovi nic posílat, protože portál sám posílá email o schválení úlohy. Veškerá zodpovědnost za uznání či neuznání vypracovaných úloh je na učiteli, který rozhoduje o úrovni žáků.

5. Nasazení koncepce v praxi

Celá koncepce byla nasazena a odzkoušena na ZŠ a MŠ Kladno, Doberská 323, která v rámci výuky poskytuje i pro celý portál technické zázemí. Portál je pro výuku plně funkční, ale některé jeho části vzhledem k časové a finanční náročnosti ani nebyly vytvářeny. Jedná se především o ochranu před útokem ze strany například hackerů a o různé zpětné kontroly zadávaných dat. Tudíž se na portál může zaregistrovat kdokoliv, i když udá špatné údaje. Vzhledem k těmto skutečnostem byl spuštěn mimo běžný port pro http komunikaci. Lze ho nalézt na adrese http://www.6zskladno.cz:8081

Celá koncepce byla nasazena od října školního roku 2013/2014 v deváté třídě. Jeho odzkoušení trvalo přibližně šest kalendářních měsíců s hodinovou týdenní dotací. Během výuky byly zařazovány i jiné tematické celky, tudíž čistého času na výuku programování bylo týdně okolo třiceti minut, zbývajících patnáct minut byl probírán jiný tematický celek či organizační záležitosti.

5.1. Způsob nasazení koncepce

Při nasazení koncepce byl dodržován doporučený didaktický postup uvedený výše. Na začátku jednotlivých témat byl kladen důraz na motivaci žáků. Bylo vysvětlováno a ukazováno konkrétní využití dané problematiky na konkrétních příkladech či větších projektech, kde je tento způsob využíván. V expoziční části hodiny byla vysvětlována konkrétní problematika sledující elektronickou oporu, ale mnohem obšírněji, než je uvedeno v textu. V některých tématech na dotazy žáků, hlavně těch, kteří se o problematiku zajímají více, bylo probíráno a vysvětlováno učivo nad rámec studijní opory. Během expoziční části hodiny, byly zařazovány prvky praktického ozkoušení dílčích částí probírané problematiky. Žáci si přímo v portálu zkoušeli možné zápisy a algoritmická řešení dané problematiky. Žáci, kteří daný den chyběli, si vždy od spolužáků či učitele zjistili, jaká konkrétní problematika byla probírána a k jaké úloze se celá skupina během hodiny dopracovala. Po nastudování opory byly s těmito žáky řešeny jen drobné konzultace na dovysvětlení nepochopené problematiky.

Druhým typem hodin po expoziční části byly hodiny, ve kterých si žáci znalosti upevňovali, aplikovali a učitel měl k dispozici zpětnou vazbu o jejich práci. Na začátku těchto hodin došlo k rychlému zopakování a shrnutí problematiky z minulých hodin a žáci vypracovávali konkrétní úlohy. Většina z nich pracovala samostatně na počítačích typu PC

popřípadě na svých smartphonech či tabletech. Vzhledem k ovládání portálu, respektive k psaní programu, jsou PC vhodnější, ale pokud se žákům pracovalo lépe na jiných zařízeních, nebylo jim bráněno. Po vypracování zadané úlohy ji žák odeslal ke kontrole. Okamžitě na učitelově počítači byl vidět výsledek práce žáka a učitel mohl ústně reagovat, což byl jediný způsob komunikace během hodiny. Po schválení úlohy to bylo žákovi ihned sděleno a on mohl pokračovat v úloze další. Jiná situace byla v případě nepřítomnosti žáka, kde naopak komunikace probíhala pouze v elektronické podobě, takže možné nedostatky ve vypracovávání úloh, byly konzultovány přes email. V případě nesnází se jedná o poměrně zdlouhavý způsob komunikace, ale ve výsledku vedl k naplnění cíle. Verifikace úloh byla ozkoušena i na mobilním telefonu, kde probíhala ještě rychleji než přes běžné PC. Tento způsob lze jen doporučit. Z pohledu učitele lze konstatovat, že díky koncepci opírajícího se o moderní technologie a on-line přístupu došlo k větší motivovanosti žáků a hlavně k lepšímu přístupu k informacím z vyučovacích hodin.

6. Výzkumné šetření v oblasti nasazení koncepce

Vzhledem k situaci, že škola danou problematiku vyučovala i v minulých letech, je možné dřívější poznatky a současné poznatky výuky s použitím nové koncepce zhodnotit výzkumem, zdali koncepce splnila předpokládaný záměr a popřípadě navrhnout její možná zlepšení. Pro výzkum, který se odehrává přímo v místě pedagogického působení s možností zařazení okamžitých změn, je převážně vhodný kvalitativní akční výzkum.

Dle některých kritiků totiž dochází jen k malému ovlivnění praxe při použití konvenčních výzkumů. Kdežto při použití akčního výzkumu se na něm aktivně podílejí jak ti, kterých se týká, v tomto případě žáci, tak i samotný výzkumník. Samotný akční výzkum by měl dodržovat tři základní pravidla. Prvním je stejné postavení jak výzkumníka, tak i zkoumaných při interpretaci výsledků. Druhým pravidlem je vztah, že zkoumaná realita souvisí s praxí. Posledním pravidlem je přenos výsledků zkoumání do reality, čímž je doporučeno, přenést poznatky do praxe a znovu zkoumat. Těmito pravidly se výzkum stává cyklickým, zúčastněným a reflektivním.[49]

Pro většinu výzkumu byl použit pro-aktivní akční výzkum, kde akce předchází sběru informací.[50] Nejprve byly provedeny změny a ty byly následně podrobeny zkoumání. Při zkoumání byly taktéž pro možnosti porovnání koncepcí použity informace z minulých let.

Mezi použité kvalitativní metody použité při tomto šetření patří dotazníkové šetření, které zkoumá názory žáků v oblasti koncepcí. Druhou použitou metodou je kvalitativní didaktický test výsledků práce žáků před použitím nové koncepce a s použitím nové koncepce. Třetí metodou je rozhovor s žáky, kteří byli vyučováni starou a novou koncepcí. Čtvrtou metodou je pozorování. Celé šetření probíhalo šest měsíců, stejně jako ověření nasazení koncepce v praxi.

6.1. Výzkumné otázky a hypotézy

Pro účely výzkumu bylo stanoveno několik základních otázek a s nimi souvisejících podotázek, které ověřují novou koncepci a její použitelnost a přínos pro praxi. Z jednotlivých otázek byly stanoveny hypotézy, které budou potvrzeny či vyvráceny a z nich vyvozeny závěry pro celou koncepci. Závěry výzkumu budou použity pro zlepšení stávajícího stavu a budou cyklicky zapracovány do další etapy výuky, která bude probíhat v budoucnu, k čemuž je akční výzkum taktéž určen.

[49] HENDL, Jan. *Kvalitativní výzkum* 1. Vydání, Praha, portál 2005 ISBN80-7367-040-2
[50] NEZVALOVÁ, D. *Akční výzkum ve škole.* In Pedagogika, 2003, roč. 53, č .3, s.300-308

otázky:

1) Vypracovávají žáci zadané úlohy lépe při výuce novou koncepcí?
    a. Je úspěšnost správného řešení samostatných úloh vyšší při výuce pomocí nové koncepce?
    b. Jsou úlohy vypracovávány syntakticky lépe?
    c. Jsou použité algoritmy při tvorbě programu lépe řešeny?
2) Je výuka jazyka JavaScript pomocí nové koncepce pro žáky bližší?
    a. Je výuka pro žáky srozumitelnější než ve staré koncepci?
    b. Jak dokáží žáci využít znalosti získané výukou programování?
    c. Je žákům předkládáno dostatečné množství příkladů?
    d. Vzbudila výuka programování v někom zájem o bližší poznávání informačních technologií?

hypotézy:

H1: Běžní žáci ZŠ dosahují lepších výsledků při řešení samostatných úloh při výuce novou koncepcí.

6.2. Metodika výzkumu

Pro účely stanovení odpovědí na výše uvedené otázky a na ověření hypotézy jsou použity kvalitativní přístupy. Při sběru a vyhodnocování dat se jedná o metody především didaktického testu, dotazníku, rozhovoru a pozorování. Všechen výzkum byl proveden na žácích devátého ročníku ZŠ. Většině žáků bylo v průběhu výzkumu 14 let.

6.2.1. Didaktický test

Pro zjištění odpovědí na otázku „Vypracovávají žáci zadané úlohy lépe při výuce novou koncepcí?“ a její podotázky sloužil především didaktický test, který byl realizován formou odevzdávaných úloh a samostatných úloh.

Pojmem „Odevzdávané úlohy“ jsou myšleny úlohy, které žáci vypracovávali v průběhu celého odzkoušení konceptu. Úlohy vypracovávali samostatně s jakoukoliv podporou. Celkem se jednalo o 35 úloh, které byly postupně hodnoceny. Hodnocení mělo pro žáky formativní charakter.

Úlohy vypracovávalo celkem 20 žáků. Celé šetření probíhalo od října 2013 do března 2014. Zadání jednotlivých úloh je uvedeno v kapitole 4.4.2. Výsledky jednotlivých úloh

byly postupně analyzovány pro zjištění kladů a záporů. Jednalo-li se o systematický klad či zápor byly vyvozeny drobné změny do navržené koncepce. Změny byly opět hodnoceny v rámci cyklického akčního výzkumu formou didaktických testů či pozorování a rozhovorů.

Pojmem „Samostatná úloha“ je myšlena práce, kterou žáci vypracovávali samostatně a to především bez pomoci jiných osob. Jednalo se o didaktické testy, které byly hodnoceny známkou, která byla podkladem pro hodnocení na vysvědčení. Hodnocení samostatných úloh mělo sumativní charakter.

Celkem bylo žákům zadáno pět úloh. První zkoumaná skupina, která se učila starou koncepcí, odevzdávala úlohy ve školním roce 2012/2013. Stejných pět úloh dostala skupina žáků učících se novou koncepcí ve školním roce 2013/2014. Počty žáků obou skupin se lišili, podle počtu přítomných v den zadávání práce. Časová období byla oba roky od října do března. Porovnáním výsledků samostatných úloh první a druhé skupiny lze usuzovat, zdali při výuce novou koncepcí dochází k lepšímu řešení samostatných úloh.

6.2.2. Dotazník

Dotazníkové šetření se především týkalo otázky „Je výuka jazyka JavaScript pomocí nové koncepce pro žáky bližší?“ a její podotázek. V dotazníku žáci celkem odpovídali na pět otázek. U otázek vybírali jednu či více odpovědí. Jednotlivé otázky a jejich odpovědi jsou uvedeny v kapitole 6.3.2.

Dotazník byl rozdán přítomným žákům po skončení výuky programování v březnu a to jak skupině učící se starým způsobem v roce 2013, tak skupině učící se novou koncepcí v roce 2014. V roce 2013 odezvalo dotazník 11 žáků a v roce 2014 17 žáků. Na vypracování dotazníku byla žákům poskytnuta doba na konci hodiny. Ke každé otázce, či na konec dotazníku mohli žáci připsat své vlastní hodnocení výuky, přínosy a nedostatky koncepce.

6.2.3. Rozhovor a pozorování

Nejpravidelnějším způsobem získávání informací týkající se nové koncepce výuky programování a algoritmizace byly rozhovory a pozorování. Pozorování, stejně tak jako rozhovory, probíhaly od října 2013 do března 2014. Jednalo se především o pozorování práce žáků při využívání nového podpůrného prostředí a jejich celkový přístup k výuce při využití nové koncepce. Rozhovory byly vedeny většinou ke konci hodiny, či po hodině

s žáky, kteří byli ochotni sdělit své názory na koncepci výuky. Při tomto způsobu získávání dat si byli v podstatě všichni účastníci výzkumu rovni a tak jak uvádí [51], všichni aktéři výzkumu se podíleli na jeho analýze. U akčního výzkumu nešel přesně oddělit sběr dat od jejich samotné analýzy. Zjištěné nedostatky byly cyklicky odstraňovány a klady byly více zdůrazněny při následné výuce.

[51] PAVELKOVÁ, A. *Akční výzkum v pedagogickém prostředí.* Brno: Masarykova univerzita 2012. Ústav pedagogických věd

6.3. Výsledky

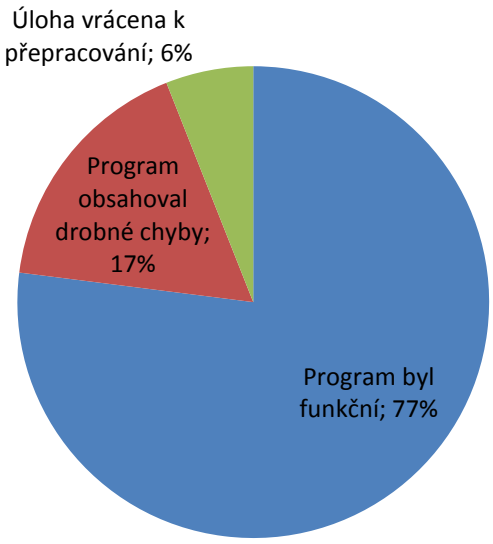
6.3.1. Zhodnocení odevzdaných úloh

V průběhu výuky každý žák odevzdal minimálně 35 úloh, které průběžně vypracovával. Každá úloha byla kvalitativně zhodnocena a buď schválena, nebo neschválena. Při schvalování nešlo vždy jen o sto procentní funkčnost, ale spíše o správný algoritmus vedoucí k řešení daného problému. Kvůli úspoře času během výuky ve škole, byly schvalovány úlohy s drobnými syntaktickými chybami.

Z vlastního pozorování, které probíhalo po celou dobu odzkoušení koncepce, lze konstatovat, že většina úloh, kterou žáci odevzdávali, byla ve funkčním stavu a byla vypracována podle zadání úlohy. Pro toto pozorování byla vypracována tabulka, která obsahovala číslo úlohy a tři možné varianty. Jednalo se o variantu odevzdáno funkční, odevzdáno s drobnými nedostatky, vráceno k přepracování.

Hodnocení úloh, jako takových bylo subjektivně zpracováváno, jako nejobjektivnější hodnotu lze použít úlohy, které byly plně funkční. Žáci při vypracovávání úloh mají k dispozici jakékoliv prostředky, jelikož se jedná o fixační a aplikační část výuky. Z výsledků pozorování vyplývá, že 77% žáků odevzdávalo funkční programy, 17% žáků odevzdalo úlohu s drobnými chybami a 6% úloh bylo vráceno k dopracování či k přepracování. Jednalo se o prvotní odevzdávání, kde uvedených 6% žáků muselo odevzdat úlohu znovu.

Vzhledem k tomu, že výuka jazyka JavaScript probíhala na základní škole i v minulých letech, je možné posuzovat i úlohy, které žáci vypracovávali při výuce jinou koncepcí. Nelze porovnávat jednotlivé úlohy, jelikož v minulé koncepci byly úlohy jiné, ale lze z pozorování vyvodit kvalitativní rozdíly. Úlohy, které žáci odevzdávali v rámci nové koncepce, byly jednoznačně funkčnější, než úlohy, které vypracovávali dřívější žáci. Ty obsahovaly mnohem více syntaktických chyb, což se dá přisuzovat nedostatečné podpoře, která byla odstraněna tím, že úlohy se vypracovávají těsně pod textem podpory. Tím se dá zodpovědět i otázka, zdali žáci vypracovávají úlohy syntakticky lépe. V případě zkoumání použitých algoritmů, lze taktéž dospět k odpovědi, že žáci volili lepší algoritmy, ale u této problematiky rozdíl mezi předešlou a novou koncepcí, tak zřetelný není. Nelze tedy jednoznačně konstatovat, že výuka pomocí nové koncepce přináší i vhodnější a přesnější algoritmy.



Graf 1: Kvalita odevzdaných úloh

6.3.2. Zhodnocení samostatných úloh

V oblasti samostatných úloh je situace pro zhodnocení žákovských výsledků příznivější z důvodu možnosti posouzení výsledků i předešlých výuk. Pro možnost porovnání byly samostatné úlohy zadány ve stejném znění, jako v předešlé koncepci. Odevzdané úlohy z minulé koncepce je tak možné posoudit s úlohami vypracovanými pomocí koncepce nové.

Pro hodnocení bylo využito celkem pět samostatných úloh. V každé z nich bylo možno posuzovat nejméně dvacet úloh od jednotlivých žáků. Pro základní hodnocení bude použito procentuální hodnocení plně funkčních úloh z dané koncepce. Jako vedlejší kritérium bude posuzována kvalita odevzdaných úloh. Při srovnávání jsou použity úlohy od různých skupin s různými studijními výsledky, které jsou dány schopnostmi, znalostmi, ale které před touto samotnou výukou prošli stejnými základy a vyučoval je stejný učitel. Dle průměrných klasifikačních výsledků, byla první skupina, která se učila starou koncepcí ve školních výsledcích úspěšnější. Nyní zkoumaná skupina, na které probíhala výuka novou koncepcí, se jeví dle průměru známek slabší.

První samostatná úloha byla zaměřena na problematiku proměnných, proměnných typu pole a jejich výpis. Zadání úlohy, tak jako všech dalších je rozkrokováno z důvodu průběžnějšího řešení úloh a uvědomění si algoritmické posloupnosti. Každá úloha je doplněna slovy a vysvětlením, co je po žácích požadováno.

Samostatná úloha 1:

1) Vytvořte proměnnou A a uložte do ní hodnotu 10.

2) Vytvořte proměnnou typu pole s názvem POLE.

3) Jako první hodnotu do POLE uložte proměnnou A.

4) Jako druhou hodnotu do POLE uložte číslo 5.

5) Jako třetí hodnotu do POLE uložte součet předcházejících hodnot.

6) Vypište součtové matematické operace a výsledek pomocí document.write.

Tabulka 1: Zhodnocení první úlohy

koncepce	odevzdáno úloh	z toho plně funkčních	procentuální úspěšnost
stará	10	7	70%
nově navržená	19	16	84%

Jak z uvedených výsledků vyplývá, úspěšnost žáků v řešení první úlohy je o trochu vyšší při výuce novou koncepcí, také odevzdané zpracování je o trochu ucelenější, ale některé chyby objevující se u odevzdaných úloh, byly totožné v obou koncepcích.

Druhá samostatná úloha byla zaměřena na problematiku funkcí s argumentem.

Samostatná úloha 2:

1) Vytvořte funkci "soucin", která bude mít 3 argumenty.

2) V prvním a druhém argumentu předáte čísla pro součin a ve třetím argumentu barvu.

3) Výsledkem funkce bude číslo, které reprezentuje výsledek součinu a bude napsáno barvou předanou ve třetím argumentu.

4) Tuto funkci zavolejte v programu 2x a to:

```
soucin(2,10,"green");
soucin(6,15,"red");
```

Tabulka 2: Zhodnocení druhé úlohy

koncepce	odevzdáno úloh	z toho plně funkčních	procentuální úspěšnost
stará	7	4	57%
nově navržená	15	10	67%

Taktéž srovnání výsledků druhé úlohy přináší lepší výsledky při výuce novou koncepcí. Z pohledu odevzdaného zpracování se úlohy jeví podobně a nelze konstatovat, které jsou lépe zpracovány.

Třetí samostatná úloha je zaměřena na problematiku větvení programu a žáci na základě vstupních hodnot proměnných volí konkrétní větev.

Samostatná úloha 3:

1) Vytvořte proměnnou TEPLOTA a do ní uložte teplotu od -50 do 50.

2) Vytvořte proměnnou OBLOHA a do ní uložte buď SLUNCE, nebo MRAKY.

3) Výsledkem programu budou 4 varianty odpovědí.

a) TEPLOTA<0 a jsou MRAKY, pak si obléknu kabát a vezmu čepici.

b) TEPLOTA<0 a svítí SLUNCE, pak si obléknu kabát a čepici si nevezmu.

c) TEPLOTA>=0 a svítí SLUNCE, pak si obléknu jen mikinu.

d) TEPLOTA>=0 a jsou MRAKY, pak si vezmu deštník.

Tabulka 3: Zhodnocení třetí úlohy

koncepce	odevzdáno úloh	z toho plně funkčních	procentuální úspěšnost
stará	12	6	50%
nově navržená	16	10	63%

I ve třetí úloze vyšly výsledky ve prospěch nové koncepce. Z hlediska ucelenosti příkladů, lze zde konstatovat lepší výsledky vypracovávaných příkladů novou koncepcí a to především z hlediska použité algoritmizace podmínek, která byla zpracována mnohem lépe.

Čtvrtá samostatná úloha měla zadání otevřenější, ale výsledek by měl být u všech podobný. Úloha je zaměřená na podmínky a cykly. Tato úloha některé žáky tak zaujala, že ji ještě doma dopracovávali a rozšiřovali.

Samostatná úloha 4:
Vytvořte program, který bude umět odpovídat na nejméně tři fráze, dokud uživatel nenapíše „konec“.

Tabulka 4: Zhodnocení čtvrté úlohy

koncepce	odevzdáno úloh	z toho plně funkčních	procentuální úspěšnost
stará	14	10	71%
nově navržená	17	14	82%

Jak se ukázalo, zadání čtvrté úlohy bylo pro žáky poměrně motivační. Byť ne všechny programy pracovaly, ale jednalo se spíše o drobné syntaktické chyby. Každý žák zvolil trochu jiný přístup, ale všichni zadávali data přes dialogové okno. Větší úspěšnost v řešení, měla opět nová koncepce. Vzhledem k rozdílným přístupům nelze porovnat a zhodnotit různá zpracování na úrovni koncepcí.
Poslední samostatná pátá úloha byla zaměřena na matematické funkce

Samostatná úloha 5:
1) Vytvořte program, který rozdělí úhel 2 PÍ (kruh) do cca 20 částí (můžeš zvolit libovolně klidně po stupni).
2) Každou část vypište na samostatný řádek a přiřaďte k ní příslušnou hodnotu sinu.
3) Příklad
SIN 2PI/20*0 je .....
SIN 2PI/20*1 je .....
SIN 2PI/20*2 je .....

Tabulka 5: Zhodnocení páté úlohy

koncepce	odevzdáno úloh	z toho plně funkčních	procentuální úspěšnost
stará	9	7	78%
nově navržená	15	12	80%

Výsledky z hlediska funkčních úloh byly u páté úlohy srovnatelné, byť nová koncepce měla nepatrně převahu. Z hlediska zpracování lze konstatovat, že u obou porovnávaných koncepcí lze pozorovat prvky vlastní implementace kódu a vylepšení úlohy, která byla studentům zadána. Opět nelze jednoznačně posoudit, jaké byly lépe zpracovány.

Po zpracování jednotlivých výsledků z úloh lze odpovědět na výše položené otázky. „Je úspěšnost správného řešení samostatných úloh vyšší při výuce pomocí nové koncepce?“ S přihlédnutím na výsledky analýzy úloh lze říci, že ano.

Této problematiky se týká hypotéza: „H1: Běžní žáci ZŠ dosahují lepších výsledků při řešení samostatných úloh při výuce novou koncepcí.“ Z výše zmíněných závěrů lze tuto hypotézu potvrdit.

6.3.3. Výsledky dotazníkového šetření

Dotazníkové šetření bylo uskutečněno v obou koncepcích s podobnými otázkami. Ve druhé koncepci se stala jedna otázka z dotazníku bezpředmětnou, tak byla nahrazena jinou a jedna otázka byla upřesněna. Celkem žáci odpovídali na pět otázek, které byly uzavřené s výběrem jedné nebo několika odpovědí.

Dotazníkové šetření je zaměřeno na druhou základní otázku týkající se kvalitativních znaků koncepcí a díky dotazníkovému šetření a rozhovoru s žáky, lze na základní a dílčí otázky nalézt odpověď.

Dotazník byl rozdán žákům, kteří prošli výukou jazyka JavaScript. Žáci měli možnost dotazník vyplnit anonymně či se mohli podepsat. Volba byla na nich. Pokud měli potřebu, mohli na zadní straně zhodnotit svůj pohled na výuku tohoto celku s jeho výhodami a nevýhodami.

První otázka se týkala srozumitelnosti výuky jazyka JavaScript. Žáci mohli vybrat pouze jednu možnost.

Otázka 1: Byla pro tebe výuka JavaScriptu srozumitelná? Byly zde uvedeny čtyři možné odpovědi.

A) Ano.

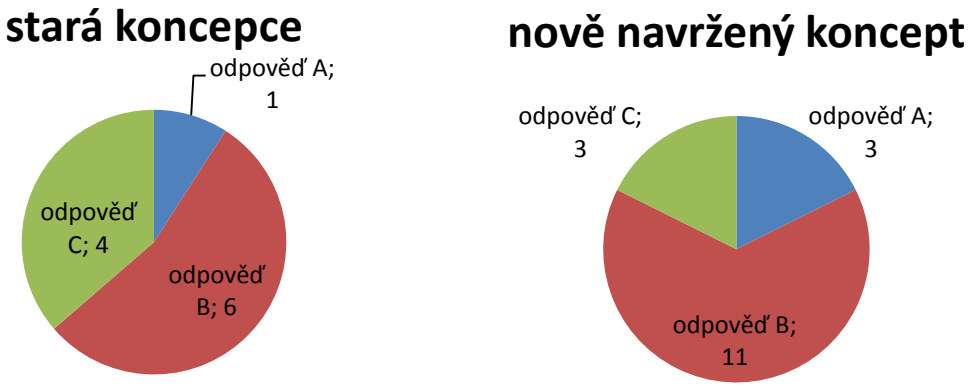
B) Spíše ano, ale občas jsem potřeboval pomoci (od učitele, kamaráda).

C) Spíše ne, ale s dopomocí (učitele, kamaráda) jsem učivo pochopil.

D) Ne, nevím, o co šlo.

Tabulka 6: Absolutní hodnoty odpovědí otázky 1

koncepce	odpověď A	odpověď B	odpověď C	odpověď D
stará koncepce	1	6	4	0
nově navržená koncepce	3	11	3	0



Graf 2, 3: Grafické zhodnocení 1. otázky

Při pohledu na procentuální vyjádření jednotlivých odpovědí, lze dospět k závěru, že žáky byla subjektivně výuka hodnocena v nové koncepci, jako srozumitelnější. Došlo ke zvýšení jak srozumitelnosti, tak i srozumitelnosti s dopomocí. Na otázku, zdali je výuka srozumitelnější než ve staré koncepci lze odpovědět ano.

Druhá otázka se týkala využitelnosti a následného používání jazyka JavaScript. Žáci mohli vybrat libovolné množství odpovědí.

Otázka 2: K čemu jsi schopen/na využít JavaScript?

A) Rozumím příkazům, ale nedokáži je použít.

B) Rozumím příkazům a dokáži je použít v programu.

C) Dokáži sestavit jednoduchý program.

D) Dokáži využít cizí JavaScript a upravit ho pro svou potřebu.

E) Dokáži sestavit složitější program.

Tabulka 7: Absolutní hodnoty odpovědí otázky 2

koncepce	odpověď A	odpověď B	odpověď C	odpověď D	odpověď E	počet respondentů
stará koncepce	1	4	7	5	2	11
nově navržená koncepce	2	9	8	5	3	17

Tabulka 8: Procentuální hodnoty odpovědí otázky 2

koncepce	odpověď A	odpověď B	odpověď C	odpověď D	odpověď E
stará koncepce	9%	36%	64%	45%	18%
nově navržená koncepce	12%	53%	47%	29%	18%

Vyhodnocení této otázky je poměrně komplikované. U některých žáků došlo k výběru sice vícero odpovědí, ale zároveň tyto odpovědi byly protikladné, tudíž validita je poměrně sporná. Významnější rozdíly se objevily u odpovědí B, C, D, kde došlo k přesunu odpovědí u nové koncepce směrem k odpovědi, že žáci rozumí příkazům a dokáží je využít v programu. U hraničních hodnot, které byly pro žáky srozumitelnější, je výsledek obou koncepcí podobný.

Na otázku, jak žáci dokáží využít znalosti získané výukou, lze odpovědět, že většina žáků vyučovaná novou koncepcí si myslí, že rozumí příkazům a dokáže je využít v programu a zároveň i sestavit jednoduchý program. Tyto dvě odpovědi často spolu korespondovali.

Třetí otázka se týkala webové podpory pro výuku jazyka JavaScript. Stará a nová koncepce měla různé otázky. Žáci mohli vybrat pouze jednu odpověď.

Otázka 3 – stará koncepce: Uvítal/a bys webovou podporu výuky?

A) Ano.

B) Je mi to jedno.

C) Ne.

Otázka 3 – nová koncepce: Byla webová podpora přínosem výuky?

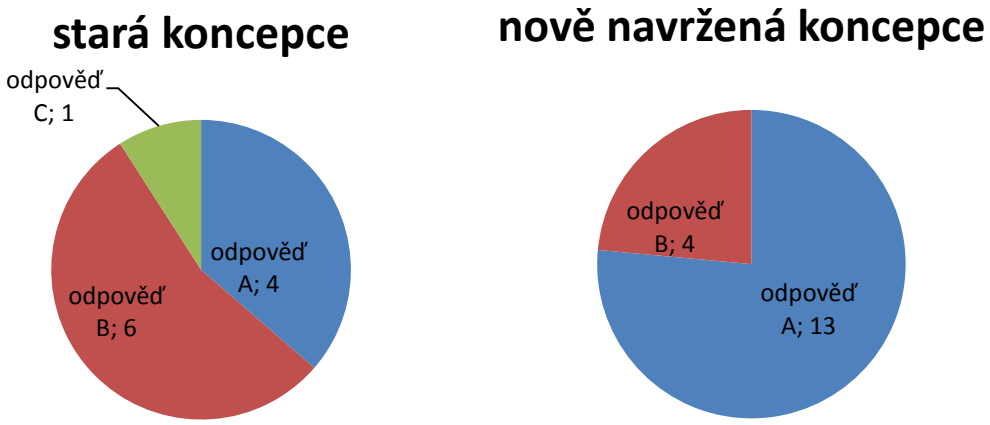
A) Ano.

B) Je mi to jedno.

C) Ne.

Tabulka 9: Absolutní hodnoty odpovědí otázky 3

koncepce	odpověď A	odpověď B	odpověď C
stará koncepce	4	6	1
nově navržená koncepce	13	4	0



Graf 4, 5: Grafické zhodnocení 3. otázky

Třetí otázka byla orientační a měla zjistit zájem žáků o výuku jazyka Javascript s webovou podporou. Jak z dotazníku vyplývá, většině žáků je asi v podstatě jedno, jak je bude učitel učit a jaké formy a metody použije. Ve chvíli, kdy jsou žáci novou koncepcí vyučováni, pak webovou podporu, na rozdíl od výuky bez podpory jako přínos vidí.

Čtvrtá otázka se týkala množství příkladů použitých pro výuku jazyka JavaScript. Žáci mohli vybrat pouze jednu odpověď.

Otázka 4: Bylo při výuce použito dostatečné množství příkladů?

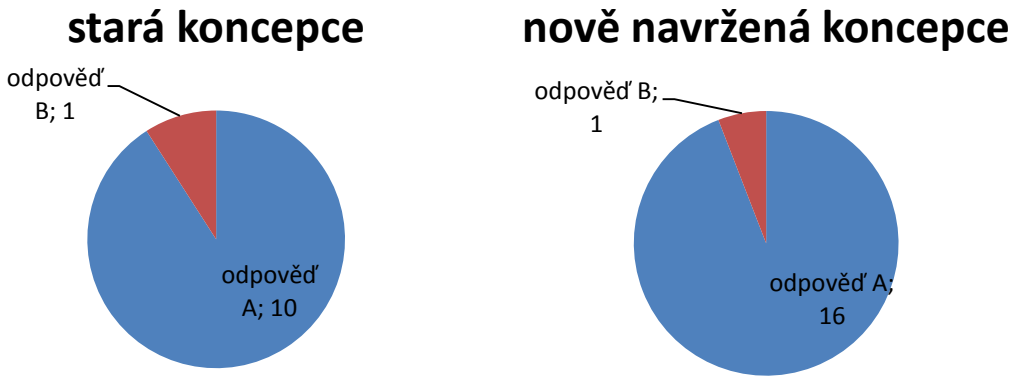
A) Ano.

B) Nevím.

C) Ne.

Tabulka 10: Absolutní hodnoty odpovědí otázky 4

koncepce	odpověď A	odpověď B	odpověď C
stará koncepce	10	1	0
nově navržená koncepce	16	1	0



Graf 6, 7: Grafické zhodnocení 4. otázky

Většina žáků jak v nové tak ve staré koncepci odpověděla, že množství příkladů na procvičení bylo dostačující. Jen jeden žák doplnil v nové koncepci k hodnotě nevím, že by uvítal příkladů ještě více. Z výsledků lze konstatovat, že odpověď na otázku, zdali je žákům předkládáno dostatečné množství příkladů, zní ano.

Poslední pátá otázka se týkala budoucí orientace žáků. Zjišťovala, zdali se chtějí v budoucnu zabývat informačními technologiemi a pokud ano, zdali je ovlivnila výuka JavaScript. Což byl doplněk otázky z loňského roku.

Otázka 5: Chceš se v budoucnu zabývat informačními technologiemi? Pokud Ano, ovlivnila tě výuka programování?

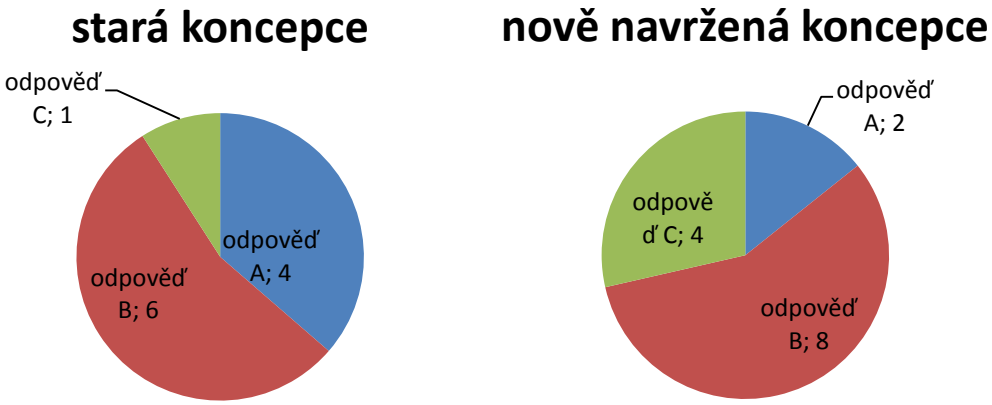
A) Ano.

B) Nevím.

C) Ne.

Tabulka 11: Absolutní hodnoty odpovědí otázky 5

koncepce	odpověď A	odpověď B	odpověď C	výuka ho ovlivnila	výuka ho neovlivnila
stará koncepce	4	6	1	X	X
nově navržená koncepce	3	8	4	2	1



Graf 8, 9: Grafické zhodnocení 5. otázky

Jak z výsledků vyplývá, ve třídě vyučované novou koncepcí je menší počet žáků, kteří by se chtěli v budoucnu zabývat informačními technologiemi, než ve třídě vyučovanou starou koncepcí. V nové koncepci odpověděli tři žáci, že by se chtěli v budoucnu zabývat informačními technologiemi a dva z nich ovlivnila i výuka programování. Jeden žák, který uvedl, že ho výuka programování neovlivnila, doplnil, že mě rád programování už před výukou. Můžeme tedy i odpovědět na poslední podotázku, zdali výuka programování v někom vzbudila zájem o informační technologie také kladně.

6.3.4. Výsledky rozhovorů s žáky a pozorování

Nejčastěji používanou formou při výzkumu byly rozhovory s žáky a přímé pozorování jejich činností. Ty probíhaly každou vyučovací hodinu popřípadě i po ní. Dílčí změny, tak jak doporučuje akční výzkum, byly ihned zakomponovány a znovu ozkoušeny v praxi.

Mezi nejčastěji zmiňovanými se objevila jednoduchost a rychlost celého systému. Žáci si pochvalovali, jak rychle lze program napsat a odzkoušet, bez jakékoliv další instalace. Další pro ně velkou výhodou bylo možnost si programy psát a zkoušet v klidu z domova a to nejen na PC, ale i dotykových zařízeních například tabletech a smartphonech.

Jako nevýhoda se z pohledu žáků jeví ne úplně snadný přístup k chybám v programu. V případě nefunkčnosti programu není na první pohled vidět, kde udělali chybu. V podstatě chybí základní ladění programu a v případě že je, tak každý internetový prohlížeč tuto problematiku řeší jiným způsobem.

Další nevýhoda, která byla v průběhu koncepce odstraněna, byly potvrzovací emaily při schválení úlohy. V původním návrhu celé koncepce nebylo počítáno s automatickým odesíláním emailů, při schválení úlohy, což se ukázalo jako prvek, který zdržoval celou práci, jak učitele, tak žáků a musel být řešen jinou cestou.

Koncepce nebyla brána jako náhražka e-learningového portálu, ale v některých situacích by žáci tuto možnost uvítali. Jednalo se především o uložení jejich úloh. Po schválení úlohy mají sice žáci přístup ke správnému řešení, ale chtěli by mít i možnost vidět své řešení, které odevzdávali. Ukládání si svých výsledků do souborů jim moc vhodné a rychlé nepřišlo.

Při procvičování žáci používali ve většině případů i podpůrný text, ze kterého čerpali syntaxi a příkazy. Práce byla z jejich strany aktivnější, než u předchozí koncepce. Mále nedostatky lze sledovat v oblasti navigačních prvků stránky, kde by chtělo zvolit možnosti automatické aktualizace stavu, například technologií Ajax s propojením na stav úloh v databázi, ale jednalo by se o větší zásah, který je možné zapracovat postupem času.

7. Závěr

Diplomová práce se zabývala problematikou programování a algoritmizace na základních školách, což je ze strany tvůrců rámcových vzdělávacích programů dosti opomíjené téma. Naštěstí existuje velká řada škol, které si v klíčových kompetencích dokáží prostor pro tuto výuku nalézt. Algoritmizace a programování jak bylo na začátku této práce uvedeno, není pouze výuka určitého tématu, ale způsob myšlení, který je v současné době informačních technologií velice přínosný a pro dobrou ekonomickou budoucnost nepostradatelný.

Práce v kapitole teoretických východisek byla právě zaměřena na problematiku výuky algoritmizace a programování v různých státech světa, ze kterých uváděla části kurikulárních dokumentů. V dokumentech se zaměřovala právě na výuku programování a algoritmizace, která je v různých státech pojímána jinak. Některé státy s touto problematikou již začínají v mateřských školách a některé se snaží tuto problematiku odsunout až na druhý stupeň. Některé státy tuto problematiku na úrovni kurikula neřeší vůbec.

U některých států byly nastíněny možnosti toho, co se žáci v určitých ročnících učí a jaké by měli mít výstupy. Tyto části byly velmi důležité pro návrh vlastní koncepce a její posloupnosti probírané problematiky.

Druhou částí teoretických východisek by se dala nazvat problematika koncepcí. Na základě její analýzy byly koncepce rozděleny do několika skupin a uvedeny jejich výhody a nevýhody. Tyto poznatky společně s vhodnými organizačními formami, metodami a didaktickými prostředky daly vzniknout nové koncepci.

V části návrhu vlastní koncepce byl brán zřetel na poměry ve škole, kde probíhalo vlastní odzkoušení. Jako vyučovaný jazyk byl zvolen JavaScript s jeho výhodami i nevýhodami. Pro výuku byl naprogramován výukový portál, který obsahuje podpůrnou část v podobě základních učebních textů a úloh na procvičování. Jedná se o řízenou výuku učitelem, kde právě on rozhoduje o tom, co se daný žák učí. Tento portál je volně dostupný a pro svou výuku jazyka JavaScript ho může použít jakýkoliv učitel. Na podobné koncepci by se dal vyučovat jakýkoliv programovací jazyk, ale musel by být řešen překlad programu nějakým emulátorem, tudíž by bylo celé prostředí robustnější a možná méně flexibilní novým prvkům.

V poslední části bylo provedeno výzkumné šetření v oblasti nového konceptu, které potvrdilo celkový přínos pro výuku programování. Nejedná se o převratný způsob výuky,

ale lze v něm spatřovat zlepšení výsledků žáků ve zpracovávaných úlohách. Dále celá koncepce směřuje k otevření programování pro všechny současné platformy bez nutnosti větších softwarových zásahů, což je pro žáky možný motivační prvek.

8. Seznam použitých informačních zdrojů

1. Do roku 2016 na západních trzích nastane nedostatek IT odborníků [online]. Businessinfo.cz [cit. 2013-12-30] Dostupný z WWW: <https://www.businessinfo.cz/cs/clanky/do-roku-2016-na-zapadnich-trzich-nastane-nedostatek-it-odborniku-31954.html>.
2. *Problematika algoritmizace.* Mendelova univerzita [cit. 2013-12-30] Dostupný z WWW: <https://akela.mendelu.cz/~mot/vyuka/skralg01.pdf>.
3. DOHNAL, Pavel. *Programování na základních školách.* Diplomová práce. [cit. 2013-12-30] Dostupný z WWW: <http://is.muni.cz/th/72886/pedf_m/Diplomova_prace_Pavel_Dohnal.pdf>.
4. TAUFEN, KOTYK, HRUBINA. *Algoritmy a algoritmizace – vývojové diagramy.* Pardubice: Univerzita Pardubice fakulta informatiky a elektrotechniky. ISBN 978-80-7395-182-5.
5. BECHYŇOVÁ, *Marta. Stránky k výuce informatiky-Algoritmizace*. [online]. [cit. 2014-05-23] Dostupný z WWW: <http://www.ivt.mzf.cz/algoritmizace-a-programovani/uvod-do-algoritmu/2-algoritmizace/>.
6. DVORSKÝ, ĎURÁKOVÁ, OCHODKOVÁ. *Základy algoritmizace*. Ostrava: Technická univerzita, 2004.
7. MARTIŠEK, Dalibor. *Algoritmizace a programování v Delphi*. 1. vyd. Brno: Littera, 2007.
8. BÉLIK, Miloš. *Malé programovacie jazyky* [online]. 18. 4. 2011.[cit. 2013-12-31] Dostupný z WWW: <http://sopusik.wordpress.com/2011/04/18/19-male-programovacie-jazyky/>.
9. MALÝ, Martin. *Jak vést děti k programování* [online]. [cit. 2013-12-30] Dostupný z WWW: <http://www.lupa.cz/clanky/jak-vest-deti-k-programovani/>.
10. SGP systems. *Výukové programovací nástroje* [online]. [cit. 2013-12-30] Dostupný z WWW: <http://www.sgpsys.com/cz/Products.asp>.
11. Imagine logo. *Imagine LOGO programování pro děti* [online]. [cit. 2013-12-30] Dostupný z WWW: <http://www.pf.jcu.cz/imagine/index.php>.
12. RESNICK, Michel. *Biography.* MIT Media Lab [online]. [cit. 2013-12-30] Dostupný z WWW: <http://www.media.mit.edu/people/mres>.
13. Scratch. Vytvářej příběhy, hry a animace Sdílej s ostatními z celého světa [online]. [cit. 2014-05-22] Dostupný z WWW: < http://scratch.mit.edu/parents/>
14. ŠALOUN, Petr. *Základy programování.* Ostrava: Vysoká škola báňská. [cit. 2014-01-01] Dostupný z WWW: <http://homel.vsb.cz/~s1a10/educ/ZP/prednasky/13-prog-jaz.pdf>.
15. KADLEC, Václav. *Učíme se programovat v delphi a jazyce Object Pascal*. Vydání 1. Praha: Computer Press, 2001. 288 s. ISBN 80-7226-245-9.
16. Embarcadero Technologies. *History of Delphi Innovation* [online]. [cit. 2014-01-02] Dostupný z WWW: <http://www.embarcadero.com/landing-pages/delphi7-to-rad-xe5>.
17. TRONÍČEK, Zdeněk. *Učebnice jazyka Java.* 30. 9. 2011. [cit. 2014-01-02] Dostupný z WWW: <http://java.cz/article/ucebnicejazykajava>.
18. ŠKULTÉTY, Rastislav. *JavaScripr programujeme internetové aplikace*. Vydání 2. Brno: Computer Press, 2004. 224s. ISBN 80-251-0144-4.
19. KOSEK, Jiří. *PHP tvorba interaktivních internetových aplikací.* Vydání 1. Praha: Grada, 1998. 492s. ISBN 80-7169-373-1.

20. *Downloads*. [online]. The PHP Group. [cit. 2014-04-01] Dostupný z WWW: <http://php.net/downloads.php>.
21. *Rámcový vzdělávací program pro základní vzdělávání.* Praha: MŠMT, 2013. [cit. 2013-12-16] Dostupný z WWW: <http://www.msmt.cz/file/29397/download/>.
22. BRDIČKA, Bořivoj. *Síťová generace podle Tapscotta* [online]. Metodický portál. 24. 11. 2008. [cit. 2013-12-17] Dostupný z WWW: < http://spomocnik.rvp.cz/clanek/11753/SITOVA-GENERACE-PODLE-TAPSCOTTA.html>.
23. TUNKROVÁ, Dana, HRBÁČEK, Jiří. *Výuka programování robotů na 1. Stupni ZŠ.* Veřejné služby IS. [cit. 2013-12-30] Dostupný z WWW: <https://is.muni.cz/repo/1078241/TunkrovaHrbacek.pdf>.
24. KOTEK, Lukáš. *Jaké programovací jazyky se používají na středních školách?* [online]. Česká škola: 27. 5. 2013. [cit. 2014-01-01] Dostupný z WWW: <http://www.ceskaskola.cz/2013/05/lukas-kotek-jake-programovaci-jazyky-se.html>.
25. Department for education. *Stránky ministerstva školství Velké Británie* [online]. [cit. 2014-05-22]. Dostupné na World Wide Web:< http://www.education.gov.uk/get-into-teaching/subjects-age-groups/age-groups/graph-description>
26. Department for education. *Stránky ministerstva školství Velké Británie ICT*]. 2013-08-02- [cit. 2014-01-11]. Dostupné na World Wide Web: <http://www.education.gov.uk/schools/teachingandlearning/curriculum/primary/b00199028/ict>.
27. Department for education. *Computing programmes of study: key stages 1 and 2*. 2013-09 [cit. 2014-01-11]. Dostupné na World Wide Web: <http://www.computingatschool.org.uk/data/uploads/primary_national_curriculum_-_computing.pdf>.
28. Department for education. *Computing programmes of study: key stages 2 and 3*. 2013-09 [cit. 2014-01-12]. Dostupné na World Wide Web: <http://www.computingatschool.org.uk/data/uploads/secondary_national_curriculum_-_computing.pdf>.
29. Computing at school. *Computer Programming in Key Stage 3*. 2009-09 [cit. 2014-02-11]. Dostupné na World Wide Web: <http://www.computingatschool.org.uk/data/uploads/CPinKS3.pdf>.
30. *Štátny vzdělávací program. Informatická výchova.* Štátny pedagogický ústav, Bratislava:2008. [cit. 2013-12-08] Dostupný z WWW: <http://www.statpedu.sk/files/documents/svp/1stzs/isced1/vzdelavacie_oblasti/informaticka_vychova_isced1.pdf>.
31. *Štátny vzdělávací program. Informatika.* Štátny pedagogický ústav, Bratislava:2008. [cit. 2013-12-08] Dostupný z WWW: <http://www.statpedu.sk/files/documents/svp/2stzs/isced2/vzdelavacie_oblasti/informatika_isced2.pdf>.
32. BLAHO, KALAŠ. *Tvorivá informatika. 1. zošit z programovania*. Vydání 3. Bratislava: SPN, 2009. 48 s. ISBN 978-80-10-01723-2.
33. BLAHO, KALAŠ. *Comenius Logo: tvorivá informatika*. Bratislava: CL Group, 1998. 157s. ISBN 80-967999-0-8
34. SALANCI, Ľubomír: *EasyLogo-discovering basic programming concepts in a constructive manner. In: Constructionist approaches to creative learning, thinking*

and education: Lessons for the 21st century. Bratislava: FMFI UK, 2010. ISBN 978-80-89186-66-2, ISBN 978-80-89186-65-5. [cit. 2014-02-16] Dostupný z WWW: <http://www.salanci.sk/EasyLogo/Paper.pdf>.
35. BELUŠOVÁ, VARGA, ZIMANOVÁ. *Algoritmy s Pascalom*. Bratislava: SPN, 2002. 48 s.
36. Australian Curriculum, Assessment and Reporting Authority, *Digital Technologies* [online]. [cit. 2014-02-19]. Dostupné na World Wide Web: < http://www.australiancurriculum.edu.au/technologies/digital-technologies/Curriculum/F-10>.
37. SKALKOVÁ, J. *Obecná didaktika*. Praha: Grada, 2007. ISBN 978-80-247-1821-7.
38. MAŇÁK, J., ŠVEC, V. *Výukové metody*. Brno: PedF MU, 2003. ISNB 80-7315-039-5.
39. MOJŽÍŠEK, L. *Vyučovací metody*. 3. vydání, Praha: SPN 1988
40. MAŇÁK, J. *Nárys didaktiky*. 3. Vydání, Brno: PedF MU, 2003. ISBN 80-210-3123-9
41. SATRAPA, P. *Perl pro zelenáče*. Praha, Neokortex 2000, ISBN 80-86330-02-8.
42. KADLEC, V. *Učíme se programovat v Delphi a jazyce Object Pascal.* Praha, Computer Press 2001, ISBN 80-7226-245-9.
43. PÍSEK, S. *JavaScript efektní nástroj oživení WWW stránek.* Praha, Grada 2001, ISBN 80-247-0014-X.
44. DOHNAL, P. *Programování – Delphi 7 Object Pascal.* Masarykova Univerzita, Brno 2009.
45. KLEMENT, KLEMENT, LAVRINČÍK. *Metody realizace a hodnocení výuky základů programování. Olomouc*: Dostál Jiří 2012. ISBN 978-80-87658-01-7
46. Klement, LAVRINČÍK. *Programování do škol.* [online]. PROŠ [cit. 2014-03-23] Dostupný z WWW: < http://www.pros.upol.cz/files/others/ucebnice-online-v3/index.htm>.
47. WS3SCHOOLS. *PHP Tutorial* [online]. [cit. 2014-03-24] Dostupný z WWW: <http://www.w3schools.com/php/default.asp>.
48. BLAHO, KALAŠ. *Imagine Logo – programování pro děti*. Brno: Computer Press 2006, ISBN 80-251-1015
49. HENDL, Jan. *Kvalitativní výzkum* 1. Vydání, Praha, portál 2005 ISBN80-7367-040-2
50. NEZVALOVÁ, D. *Akční výzkum ve škole.* In Pedagogika, 2003, roč. 53, č .3, s.300-308
51. PAVELKOVÁ, A. *Akční výzkum v pedagogickém prostředí.* Brno: Masarykova univerzita 2012. Ústav pedagogických věd