Univerzita Karlova
Pedagogická fakulta
Katedra informačních technologií a technické výchovy

BAKALÁŘSKÁ PRÁCE

Programovací nástroje pro algoritmizaci a programování na ZŠ

Programming tools for development of algorithms and programming at elementary school

Jan Dümont

Vedoucí práce: Ing. Jaroslav Novák, Ph.D.
Studijní program: Specializace v pedagogice
Studijní obor: Informační technologie se zaměřením na vzdělávání

2019

Prohlašuji, že jsem bakalářskou práci na téma „Programovací nástroje pro algoritmizaci a programováni na ZŠ“ vypracoval pod vedením vedoucího bakalářské práce samostatně za použití v práci uvedených pramenů a literatury. Dále prohlašuji, že tato bakalářská práce nebyla využita k získání jiného nebo stejného titulu.

V Litoměřicích dne 7.7. 2019

…… podpis

Rád bych na tomto místě poděkoval vedoucímu práce, Ing. Jaroslavu Novákovi, Ph.D., za jeho cenné rady, odbornou pomoc a velkou trpělivost při vedení bakalářské práce. Dále bych chtěl poděkovat své přítelkyni Lence za nekonečnou trpělivost během psaní této práce.

ANOTACE

Bakalářská práce je zaměřena na vzdělávací oblast algoritmizace a programování na základních školách. Hlavním cílem je přinést přehled a popis konkrétních výukových metod a programů, které napomáhají k rozvoji informatického myšlení u žáků. Součástí práce je analýza výzkumů, které mapují stav výuky algoritmizace a programování na základních školách v letech 2006-2007 a v roce 2013, uskutečněných na Katedře informačních technologií a technické výchovy Pedagogické fakulty Univerzity Karlovy. Práce také obsahuje vymezení základních pojmů vztahujících se k tomuto tématu a ukázku pracovních listů ke dvěma konkrétním metodám.

KLÍČOVÁ SLOVA

Algoritmus, informatické myšlení, propedeutika, programovací jazyk, robotická programovatelná hračka

ABSTRACT

The bachelor thesis is centred on an educational area algorithm development and programming at basic schools. The main aim is an overview and a description of teaching methods and programmes which promote the development of student´s computional thinking. The thesis contains an analysis of researches which describes the situation of teaching algorithm development and programming at basic schools in 2006-2007 and in 2013. The analysis was realized by Department of Information Technology and Education at Faculty of Education at Charles University. Next part is a definition of basic concepts relating to this topic and at the end there are two examples of worksheets prepared for concrete methods.

KEY WORDS

Algorithm, Computational Thinking, Propaedeutic, Programming Language, Robotic Programmable Device

OBSAH

1 Úvod

Dle závěrů z několika výzkumů, o kterých práce pojednává, je *programování a algoritmické myšlení* považováno za okrajovou až zanedbatelnou informačně technologickou kompetenci. Na těchto závěrech se shodují dotazovaní učitelé informačních technologií na základních školách stejně jako dotazovaní žáci.

S blížícím se termínem završení nového konceptu vzdělávání informaticky zaměřených předmětů, který je podrobně rozpracován ve vládní studii nazvané *Strategie digitálního vzdělávání do roku 2020*, bude nezbytné rozšířit získávané informace také o základní poznatky a dovednosti z oblasti algoritmizace a programování. Takto získané kompetence budou dále rozšiřovány v učivu na střední škole.

Pro získání prvních poznatků v oblasti *algoritmizace a programování* je vhodné využít některý z mnoha dostupných propedeutických programovacích jazyků, případně využít různých her nebo robotické techniky, kde lze kompetence algoritmického myšlení nacvičovat.

Práce je zaměřena na představení a porovnání jednotlivých propedeutických jazyků a prostředí, které jsou na našem trhu k dispozici, jak ve formě placeného softwaru, tak ve formě volně dostupného desktopového softwaru nebo online prostředí. Součástí práce je ukázka pracovních listů a metodiky pro učitele informatiky.

2 Cíle práce

Bakalářská práce se zabývá vybranými metodami výuky *algoritmizace a programování* na základní škole a je koncipována jako pomůcka učitelů informatiky při zavádění výuky algoritmického myšlení a programování na ZŠ.

Hlavním cílem je zmapování dostupných nástrojů pro rozvoj algoritmického myšlení na ZŠ. Dílčím cílem je analýza tehdejšího stavu výuky algoritmizace a programování na základní škole. Podkladem jsou informační zdroje z výzkumů o stavu výuky algoritmizace a programování na základních školách uskutečněných na Katedře informačních technologií a technické výchovy Pedagogické fakulty Univerzity Karlovy. Dílčím cílem je analýza pojmů algoritmizace a programování.

Práce reflektuje závěry vládní směrnice Strategie digitálního vzdělávání do roku 2020, kde jsou výuka programování a rozvoj algoritmického myšlení zařazeny jak do výuky informatiky, tak do výuky ostatních předmětů. Programovací nástroje jsou porovnávány na základě vybraných kritérií, kterými jsou věková kategorie žáka, vhodnost dané metody a její vliv na rozvoj kompetencí zaměřených na algoritmické myšlení, a zda je vypracována pedagogická podpora ve formě metodického portálu či fóra.

Bakalářská práce je rozdělena na dvě části. První část se zabývá vymezením hlavních pojmů a seznámením se s výzkumy zabývajícími se výukou informatiky na základních školách. Ve druhé části jsou představeny konkrétní propedeutické programovací jazyky a ukázka pracovních listů ke dvěma konkrétním metodám.

3 Teoretická východiska

Dříve než budou uvedeny konkrétní zástupci propedeutických programovacích jazyků a prostředí, kterých lze využít při výuce algoritmizace a programování, je nezbytné vysvětlit několik pojmů, které se k dané problematice vztahují. V podkapitole 3.1. jsou vysvětleny pojmy algoritmus a algoritmizace, vlastnosti algoritmu a následné využití v několika vědních oborech i běžném životě.

Vzhledem k zaměření práce jsou podrobněji rozebrány algoritmy využívané v každodenní praxi, aby si žáci uvědomili, že algoritmy používají v mnoha činnostech, které během dne vykonávají, aniž by si to uvědomovali. Této problematice je věnována podkapitola *Algoritmy v běžném životě*, ve které jsou uvedeny konkrétní příklady používaných algoritmů.

3.1 Vymezení hlavních pojmů

3.1.1 Algoritmus a algoritmizace

Slovo algoritmus bylo až do počátku 20. století spojováno převážně s oborem aritmetiky a diferenciálních počtů, ve kterých se popisovaly postupy či návody při konkrétních početních úkonech. (Pavlíček 2012, s.19) Až na začátku 20. století se začal pojem algoritmus úzce spojovat s využitím výpočetní techniky, která díky soustavě rutinních instrukcí vykonává příslušné operace. Nové pojetí pojmu je spojováno s Alanem M. Turingem, autorem matematické definice, jež se stala ekvivalentem počítačového algoritmu, a to v době, kdy počítače prakticky neexistovaly. (Pavlíček 2012, s.20)

V následujících podkapitolách je popsáno použití algoritmů jak v různých vědních oborech, tak při každodenním používání v běžném životě. Málokdo si uvědomuje, že k využívání algoritmů není nezbytně nutné používat výpočetní techniku.

Algoritmus lze definovat mnoha způsoby.

> *„Algoritmus je schematický postup pro řešení určitého druhu problémů, který je prováděn pomocí konečného množství přesně definovaných kroků.“ (Kvasil 1984)*

> *„Algoritmus je přesný postup, který popisuje řešení daného problému v konečném počtu kroků.“ (Pšenčíková 2007)*

Vzhledem k obecnému pojetí definice algoritmu, která je však dostačující pro účely prvotní představy o algoritmu, je pro úplnost uvedena ještě klasická programátorská definice.

> *„Algoritmus je přesně určená konečná posloupnost instrukcí, jejichž realizace nám umožní pro přípustné hodnoty vstupních dat nalézt po konečném počtu kroků správná výstupní data.“ (Wirth 1989, cit v Havelková 2017, s. 5)*

Vlastnosti algoritmů pro počítačové prostředí

Přestože je algoritmus spojován s návodem, podle kterého musí člověk postupovat k dosažení kýženého cíle, jak je uvedeno v následující podkapitole, lze najít mezi ním a počítačovým algoritmem jisté rozdíly. Největším rozdílem je složitost algoritmu, jež řeší míru alternativ podobných problémů. Algoritmy vykonávané člověkem jsou mnohem jednodušší, neboť je nutné daný algoritmus pochopit a v reálném čase také provést. Naproti tomu je počítačový algoritmus schopen zpracovávat mnohem více informací než lidský mozek, a lze do něj zapracovat více alternativních postupů na základě vstupních parametrů. Přesto i zde je nutné dodržovat jisté podmínky.

Dle Pšenčíkové (2007) je třeba zajistit dodržení několika základních pravidel, která musí počítačový algoritmus splňovat:

- ***Determinovanost***: v každém kroku algoritmu musí být zcela jednoznačně určen krok následující.
- ***Obecnost***: abychom mohli daný postup označit jako algoritmus, nesmí se omezovat pouze na řešení jednoho specifického problému, ale jeho aplikace musí být použitelná pro obdobné problémy.
- ***Resultativnost***: každý algoritmus by po dosažení určitého počtu kroků měl vést k výsledku.

- ***Konečnost***: algoritmus řešící určitý problém má konečný počet kroků, které se odvíjejí od složitosti struktury daného algoritmu a počtu vstupních dat. Existují algoritmy, které najdou řešení problému po několika málo krocích, stejně tak jako algoritmy, kde řešení problému může zabrat delší časový úsek a algoritmus musí po určitém počtu kroků skončit.

Uvedená pravidla neplatí v případě objektivně orientovaných programů. Důvodem je skutečnost, že další kroky řídí uživatel ovládáním programu, a nejsou proto jednoznačně dány.

Algoritmizací v užším slova smyslu je myšlen cílený postup tvorby algoritmů, jež jsou následně zpracovány výpočetní technikou. (Bromová 2012, s. 5)

Celý proces je rozdělen do pěti fází:

- ***Formulace problému –*** je první fází, ve které si stanovíme požadavky k řešení problému na základě vložených vstupních a výstupních hodnot požadovaných přesností výsledku.
- ***Analýza problému –*** je druhou fází algoritmizace, ve které na základě množství vstupních hodnot musíme rozhodnout, zda je problém řešitelný. V případě, že je problém řešitelný, snažíme se najít nejjednodušší řešení.
- ***Sestavení algoritmu pomocí formálního zápisu*** – je třetí fází, ve které naše řešení problému z druhé fáze přepíšeme do symbolické formy pomocí přesného pořadí dílčích operací, které vedou k řešení daného problému. Tento krok však zatím neřeší problém samotný, pouze naznačuje možný postup.
- ***Vytvoření programu pomocí programovacího jazyka –*** je předposlední fáze procesu algoritmizace, která převádí daný postup do konkrétního programovacího jazyka.
- ***Odladění programu –*** je poslední fází procesu, ve kterém dochází k následným odstraňování programových chyb.

Tyto chyby jsou podle Bromové (2012, s. 5) děleny do dvou základních skupin.

- ***Syntaktické chyby*** jsou méně závažnou formou chyb, ve kterých dochází k provinění proti řádnému zápisu určitého příkazu či procedury, způsobených překlepem nebo chybějící částí dílčí části zápisu. Syntaktické chyby jsou snadno odhalitelné díky překladači, který na podobné prohřešky upozorní, a zároveň označí jejich umístění v programovém kódu.
- ***Funkční chyby*** patří mezi kritické chyby, ke kterým dochází z důvodu špatné nebo nedostatečné analýzy problému a následnému sestavení vhodného algoritmu. Tento druh chyb se velmi těžko odhaluje, jeho výskyt se projevuje neočekávanými výstupními hodnotami, které nebyly v původním algoritmu stanoveny.

Hlavním cílem výuky algoritmizace a programování přitom není naučit děti programovat v konkrétním programovacím jazyku, ale naučit je algoritmicky přemýšlet a daný problém analyzovat, tedy umět rozdělit danou úlohu na menší podúlohy, které žák dokáže vyřešit. (Pitner 2017)

Algoritmy v běžném životě

Jelikož je pojem algoritmus převážně spojován s užitím výpočetní techniky, většině lidí přijde jejich tvorba velmi složitá a mají ji spojenou s odborným informačně technologickým vzděláním. Lidé, kteří zastávají tento názor, si však neuvědomují, že algoritmy ve svém každodenním životě běžně používají, jen bez využití výpočetní techniky. Jedná se o algoritmy v pasivní formě, kdy je nutné dodržovat předepsané kroky tak, aby bylo dosaženo kýženého výsledku. Tyto návody jsou rozděleny do předem definovaných kroků v určité posloupnosti, kterou je nutno zachovat pro opakované dosažení stejného cíle.

Typickým příkladem algoritmu, ke kterému není zapotřebí výpočetní techniky, je kuchařský recept na přípravu jídla, návod na sestavení nábytku stavebnicového charakteru či servisní manuál popisující postup opravy různých technických problémů.

Při použití výše zmiňovaných algoritmů je předpokládaným vykonavatelem člověk. Z tohoto důvodu mohou být součástí takového algoritmu i příkazy, které nejsou jednoznačné a elementární. Autor algoritmu, s ohledem na inteligenci člověka,

předpokládá jisté elementární všeobecné znalosti, a proto není nutné dílčí úkoly algoritmu blíže specifikovat. Díky subjektivnímu přístupu při vykonávání příslušných kroků lze dosáhnout jisté variability výsledku.

Pokud je v návodu přikázáno „zašroubujte vrut A do otvoru B“, není třeba fázi šroubování blíže rozepisovat, tedy psát „uchopte šroubovák do pravé ruky“ (případně do levé, dle pravolevé orientace uživatele), „nasaďte hrot na drážku, kterou naleznete na hlavici vrutu“. Tento přístup by velmi prodlužoval jednotlivé postupy a do jisté míry by zbytečně tříštil podstatné části daného algoritmu.

Dalším příkladem může být kuchařský recept na pečení sušenek, kdy je u přípravy těsta napsáno: „Do těsta přidejte cukr či sůl a důkladně prohněťte“. Volba použití příslušného dochucovadla je na uvážení uživatele a záleží na druhu využití. Není třeba v receptu dodávat, že pokud chceme podávat sušenky ke kávě, měli bychom těsto osladit a sůl zvolit v případě pečení sušenek podávaných např. k vínu.

Výše zmiňované pravidlo nelze použít při tvorbě algoritmu zpracovávaného strojem, neboť v tomto případě se nepředpokládá samovolné ovlivňování postupu. Automat na pečení sušenek bude mít speciální algoritmus pro výrobu slaného pečiva, který bude odlišný od algoritmu na výrobu sladkých sušenek. Na rozdíl od člověka musí mít automat striktně zadány všechny ingredience, které je třeba do těsta přidat.

Algoritmy v matematice

V předchozí kapitole bylo ukázáno využití algoritmu v informaticky zaměřeném předmětu. Této kompetence lze využít také v neinformaticky zaměřených předmětech. Existuje předmět, který je díky svému pojetí s algoritmizací úzce provázán, a tím je matematika.

S algoritmy v matematice se setkal každý absolvent základní školy, jelikož většinu elementárních dovedností lze převést na posloupnost předepsaných instrukcí. Ukázkovým příkladem tohoto postupu je znalost násobení dvou přirozených čísel, jež lze převést na problematiku vícenásobného sčítání, které po provedení konečného počtu kroků vede ke stejnému výsledku. (Pavlíček 2012, s.19) Dalším příkladem jsou soustavy lineárních rovnic, při jejichž řešení je možné využít dva způsoby. Řešitel může použít

vlastní intuici a vhled do problému, což předpokládá znalost a pochopení dané problematiky. Při druhé možnosti může být rovnice vyřešena pomocí předepsané posloupnosti kroků, která dovede řešitele k výsledku i bez pochopení principu.

Z pohledu matematiky lze podle Galby a Pavlíčka (2012, s. 19) algoritmus charakterizovat jako návod pomocí následujících znaků:

- Je složen z konečného počtu přesně definovaných instrukcí, zapsaných konečným počtem symbolů.
- Řeší většinou celou třídu problémů v závislosti na vstupních datech.
- Může být prováděn i člověkem, který má k dispozici tužku a papír.
- K provedení algoritmu není potřeba lidská inteligence s výjimkou té, která je nutná k pochopení instrukcí.

3.1.2 Informatické myšlení

V zahraničí je pojem „Computational Thinking“ (dále už jen CT) úzce spojován se samostatným vyučujícím předmětem „Computer Science“, jež představuje zcela nový přístup k výuce informatiky. Doslovný český překlad tohoto termínu, který by zněl „výpočetní myšlení“, je srozumitelněji překládán jako „informatické myšlení“. V kapitole je nový termín podrobněji vysvětlen.

Hned na začátek je třeba uvést častou představu, že informatické myšlení je pouze doménou profesionálních programátorů, a tudíž má smysl jen u žáků, kteří se takovému povolání chtějí v budoucnu věnovat. Uvedená, ne zcela správná, představa plyne z neznalosti nebo nepochopení uvedeného termínu.

Informatické myšlení, stejně tak jako samotná algoritmizace, nemá za cíl vychovávat budoucí programátory, ale naopak má rozvíjet myšlenkové procesy při řešení problémů. Hlavní náplní tohoto procesu je naučit děti přemýšlet systémově. Popsaný přístup však není ničím převratným, neboť hodně lidí ho využívá nevědomě při každodenní aktivitě. Příkladem může být *„maminka, která ukládá zakoupené jogurty do lednice do prioritní fronty dle data spotřeby“* (Lessner 2018, s. 1).

„Žák při balení školní brašny také projevuje informatické myšlení.“ (Lessner 2018, s. 1). Dalším příkladem je *„výběr nejrychlejší fronty v obchodě využívající modelování výkonnosti.“* (Lessner 2018, s. 1).

Díky tomuto trendu by se měla změnit koncepce výuky informatiky od pouhého zvládání užití ICT ke kultivaci myšlení. K tomuto procesu by však nemělo docházet pouze při výuce informatiky, ale také v jiných předmětech, např. v matematice, a to v tématech, které s podobným myšlenkovým pojetím úzce souvisí.

Poprvé použil pojem „informatické myšlení“ Saymour Papert, autor původního programovacího jazyku Logo, když přiblížil možnost využití ICT ve výuce matematiky.

Diskuzi o informatickém myšlení rozšířila Jeanett Wingová, když ho přirovnala k základní gramotnosti, stejně jako čtení, psaní a počítání, a zdůraznila tak jeho potřebu ve výuce.

> *„CT jsou myšlenkové postupy zapojené při takovém formulování problémů a jejich řešení, které umožňují tato řešení efektivně provést agentem zpracovávajícím informace.“ (Wing 2010).*

Tímto agentem Janett Wingová rozumí stroj i člověka. Vzhledem k faktu přílišně obecného vymezení J. Wingové není tato definice vhodná pro aplikaci na školní prostředí.

Mnohem přesnější a konkrétnější definici přináší International Society for Technology in Education (ISTE) a Computer Science Teachers Association (CSTA) (Stephenson 2011).

CT dle Lessnera (2018, s. 3) je postup řešení problému, který zahrnuje mimo jiné následující charakteristiky.

- *„Formulovat problémy způsobem, který umožňuje jejich strojové řešení.“*
- *„Logicky uspořádat a zkoumat data.“*
- *„Reprezentovat data prostřednictvím abstrakcí, jako jsou modely a simulace.“*
- *„Automatizovat myšlení pomocí algoritmického myšlení (jako posloupnost kroků).“*

- *„Odhalit, prozkoumat a provést možná řešení s cílem odhalit nejúčinnější kombinaci činností a zdrojů.“*
- *„Zobecňovat a přenášet tento postup řešení problémů do dalších nejrůznějších oblastí.“*

Nedílnou součástí, bez kterých by se CT nemohlo rozvíjet, jsou také jisté předpoklady a postoje. (Lessner 2018, s. 3)

- *„Sebejistota tváří v tvář složitosti.“*
- *„Vytrvalost při řešení obtížného problému.“*
- *„Snášení nejednoznačnosti.“*
- *„Schopnost vypořádat se s otevřenými problémy.“*
- *„Schopnost dorozumět se a spolupracovat s ostatními při dosahování společného cíle.“*

Jednoduchou definici, dalo by se říci nevědeckou, přináší společnost Google, která v této oblasti vytváří vlastní web. (Lessner 2018, s. 3)

> *„CT zahrnuje sadu technik a dovedností k řešení problémů, které při psaní běžně používaných aplikací (vyhledávání, email, mapy), používají softwaroví inženýři. CT je nicméně využitelné téměř v jakémkoliv předmětu.“ (Lessner 2018, s. 3)*

Součástí CT je zejména:

- *„rozklad problému,*
- *rozpoznávání vzorů (např. v grafech na burze, ale i v procesech),*
- *zobecňování vzorů (tedy vytváření abstraktních modelů),*
- *navrhování algoritmů.“*

3.1.3 Tvořivé myšlení

> *„Tvořivost je zvláštní soubor schopností, které umožňují uměleckou, vědeckou nebo jinou tvůrčí činnost. Ta se projevuje jako vynalézavost, jako vznik nového, originálního díla nebo myšlenky, popř. tvůrčím řešením problémů.“ (Tvořivost 2018).*

Tento tvůrčí proces lze podle Žáka (2004), který čerpal z primární teorie formulované sociálním psychologem a ekonomem Grahamem Wallasem, rozdělit do čtyř fází:

- *Příprava* – v této fázi dochází k pojmenovávání, analyzování a zpracování problémové situace.
- *Inkubace* – řešitel získává od problému odstup. Oproti předchozí fázi, kde hrála roli spíše logika (analýza, zpracování), se do procesu zapojuje intuice. Řešitel získává od problému vědomě odstup, zpracovává získané informace. Díky tomuto odstupu je analytické myšlení odsouváno do podvědomí, kde dochází k hledání řešení. Během toho procesu vznikají různé nekonkrétní varianty.
- *Iluminace* – během této fáze, která je rovněž intuitivní, se z řešitele stává objevitel, což znamená, že řešitel dospěje ke konkrétnímu nápadu, jak daný problém vyřešit.
- *Verifikace* – je poslední fáze procesu, při které dochází k ověření tohoto řešení vůči původní problémové situaci, a zda ho lze k řešení použít.

Podle popisu by se mohlo zdát, že tento proces je časově náročný, a tudíž pro výuku zcela nepoužitelný. Je třeba si uvědomit, že ve většině případů je čas řešení daného problému přímo úměrný složitosti problému. Důležitou součástí tohoto procesu je analýza problému, která je základem jakéhokoliv řešení.

3.2 Informační výchova na ZŠ

Informaticky zaměřený předmět je jedním z mála předmětů, který se průběžně mění podle vývoje nových technologií, a to jak díky vývoji výpočetní techniky, tak vývoji nových softwarových prostředí. Výuka by měla reagovat ve stejné míře na vývoj technologický i sociální. Bez výpočetní techniky si dnes již nedokážeme sociální interaktivitu vůbec

představit, což společně s mnoha výhodami přináší také jistá rizika. Proto by mělo být jedním z cílů informační výchovy tento dynamický trend následovat.

Z těchto důvodů proběhly v letech 2006-2007 a 2013 rozsáhlé průzkumy, analyzující trendy výuky informatiky na základní škole. V následujících kapitolách budou shrnuty dílčí závěry mapující tehdejší vnímání role informatiky na základních školách, které z obou výzkumných šetření vyplývají. Postavení informatiky jako vyučovacího předmětu a popisu hlavních tématických celků je věnována vždy jedna podkapitola u každého výzkumu.

3.2.1 Algoritmizace a programování na ZŠ v roce 2007

V letech 2006 a 2007 proběhl za podpory MŠMT rozsáhlý výzkumný projekt mapující stávající situaci informačně technologické výchovy na základních školách pod pracovním označením VIV06. (Rambousek et al., 2007)

Hlavním cílem tohoto výzkumu bylo poznat aktuální stav v oblasti vzdělání, která na rozdíl od jiných vzdělávacích oblastí vykazuje rychlý vývoj, kterému je třeba se neustále přizpůsobovat. Explorativní metodou tohoto výzkumu bylo zvoleno dotazníkové šetření, které bylo předloženo pedagogům informaticky zaměřených předmětů na základních školách.

Kapitola je zaměřena na dílčí výsledky tohoto dotazníkového šetření. Předem musí být zmíněn zásadní fakt tohoto šetření, na který v úvodu své práce upozorňují sami autoři. Tím faktem je skutečnost, že navzdory značnému zkoumanému vzorku nebylo možné zajistit plnou reprezentativnost vzorku srovnatelnou s náhodným výběrem[1]. Z tohoto důvodu je při vyhodnocování výsledků dotazníkového šetření nutné pamatovat na skutečnost, že výsledky tohoto šetření lze vztahovat k danému vzorku respondentů, avšak k zobecňování těchto závěrů by mělo docházet velice opatrně.

[1]Základní charakteristika výzkumu. [ZPRACOVAL KOLEKTIV ŘEŠITELŮ POD VEDENÍM VLADIMÍRA RAMBOUSKA]. *Výzkum informační výchovy na základních školách*. Plzeň: Koniáš, 2007, s. 215-238. ISBN 8086948102.

Primární zaměření informační výchovy

Z výzkumu informační výchovy na základních školách realizované v letech 2006 až 2007 vyplývá několik závěrů.

Informační výchova na základních školách je realizována v rámci povinného a povinně volitelného informaticky zaměřeného předmětu, zájmových aktivit a ostatních vyučovacích předmětů. Z výzkumu dále vyplývá, že v zastoupení při rozvíjení informačně technologických kompetencí se největší měrou podílí neinformaticky zaměřené předměty (Rambousek et al., 2007).

> *„Polovina učitelů informatických předmětů deklaruje, že jejich informačně technologické kompetence jsou na úrovni, kterou sami ještě pokládají za minimální ještě přijatelnou pro kvalitní realizaci výuky informatických předmětů na základní škole. Třetina učitelů pak v tomto smyslu dokonce deklaruje, že jejich kompetence této minimálně akceptovatelné úrovně nedosahují.“ (Rambousek et al., 2007, s. 14)*

Výzkum deklaruje, že obsah a pojetí informačního vzdělávání na ZŠ je do značné míry ovlivněno právě informačně technologickými kompetencemi pedagogů. Proto je výuka převážně zaměřena na uživatelské dovednosti ovládání počítače a tvorbu digitálních dokumentů, jelikož v této oblasti dosahuje převážná většina pedagogů potřebné úrovně. Z výuky informatiky jsou eliminována témata, která jsou považována za náročnější a také ta, ve kterých nemají pedagogové dostatečné znalosti. (Rambousek et al., 2007)

Jedním z takto dotčených tematických celků je právě oblast algoritmizace a programování, která byla většinou pedagogů řazena až na poslední místo v seznamu tematických oblastí informačních a komunikačních technologiích (dále jen jako ICT) Rámcově vzdělávacího programu pro základní vzdělávání (dále jen jako RVP ZV). Většina z nich považuje tuto oblast pro úroveň základního vzdělání za příliš náročnou a odsouvá jí na úroveň středoškolského nebo vysokoškolského vzdělání (Rambousek et al., 2007, s. 102-103).

3.2.2 Algoritmizace a programování na ZŠ v roce 2013

V letech 2012–2013 proběhl výzkum *Informačně technologické kompetence dětí a jejich rozvoj na základních školách*, jehož hlavním cílem bylo zmapování aktuálního stavu a struktury výuky informačně technologických předmětů. Tento projekt volně navazuje na výzkum vedený v letech 2006-2007, jehož závěry směrem k oblasti algoritmizace byly shrnuty v předchozí kapitole.

Přestože je hodnocení zaměřeno na výsledky týkající se tematického celku *Algoritmizace a programování*, jsou zmíněny i souhrnné výsledky mapující celkový stav vzdělávací oblasti.

Primární zaměření informační výchovy

V rámci výzkumu byla shromážděna data od 1183 učitelů informaticky zaměřených předmětů na základní škole z celkového počtu 3500 náhodně oslovených základních škol. (Černochová 2013, s. 7), přičemž genderové rozdělení pedagogů bylo přibližně rovnocenné.

Na rozdíl od předchozího výzkumu se posiluje pozice povinné informatiky, coby dominantní aktivity, ve které je rozvíjena informační gramotnost žáků. Nicméně ani ostatní neinformatické předměty, ve kterých se při výuce využívá informačních technologií, nejsou v tomto ohledu zanedbatelné.

Stávající koncepce oblasti ICT je v RVP ZV zastaralá a nereflektuje výrazný pokrok v této oblasti vzdělání. Chybí zde jasně daná koncepce výuky a vše je příliš obecné. (Černochová 2013)

Je požadována restrukturalizace hodinových dotací pro informaticky zaměřené předměty, případně začlenění příbuzných témat do učiva ostatních předmětů.

Výzkum také potvrdil závěry z roku 2006 ohledně náplně a obsahu informatických předmětů, který se v převážné většině základních škol redukuje na uživatelské dovednosti, které zahrnují práci se základními kancelářskými aplikacemi, práci s vyhledávačem a povědomí o právních a etických aspektech při práci s informacemi.

Nové šetření opětovně potvrzuje tendenci odsouvat algoritmizaci a programování až do středoškolského vzdělávání. V malé míře se zvýšil poměr pedagogů s alespoň minimální nebo vyšší informaticky technologickou kompetencí, která je přijatelná pro realizaci výuky. Pouze pětina dotázaných pedagogů označila své ICT kompetence za nedostatečné k realizaci výuky. Také se ukazuje nový trend multiplatformní výuky, který reflektuje na současné možnosti, bohužel je užíván především při práci s kancelářskými aplikacemi.

3.2.3 Vývoj algoritmizace a programování na ZŠ do roku 2020

Jelikož je informatický obor jedním z nejvíce rozvíjejících se vědních oborů, musí na tento trend reagovat také školní informatika, jejíž postavení mezi povinnými předměty základního vzdělání se v průběhu času mění. Následující kapitola shrnuje stávající stav informačně zaměřených předmětů a popisuje vizi MŠMT, jejíž plnění by mělo v budoucnosti vést ke změně výuky informatiky.

Současný stav informační výchovy

Povinný předmět práce na počítači, informatika či alternativní název, který si škola zvolila v rámci svého školního vzdělávacího programu (dále jen ŠVP), a který musí reflektovat (rámcové) vzdělávací cíle dané v RVP ZV, se vyučuje jen 45 minut týdně, a to jeden rok na prvním, a jeden rok na druhém stupni. Zařazení povinně volitelného předmětu s obdobným zaměřením je na uvážení vedení školy, nikoliv však povinnost. Během této nízké hodinové dotace si žáci většinou osvojí pouze základní uživatelsky zaměřené dovednosti – tvorba textu, tabulky a prezentace. (Černochová 2013)

Přestože oblast Informační a komunikační technologie je zakotvena v RVP ZV, zaměření této oblasti se nezměnilo od roku 2004, což je považováno za nedostatek. Z výše zmiňovaného důvodu se RVP ZV zaměřilo pouze na schopnost vyhledávat informace na internetu a vytvářet dokumenty v běžně dostupných kancelářských balíčcích. Nic ale nebránilo tomu vyučovat informatiku směrem k tvůrčím přístupům a algoritmizaci. Přesto byla oblast zaměřená na algoritmizaci a algoritmické myšlení naprosto opomenuta, z čehož vyplývá skutečnost, že musel existovat ještě jiný a nejspíše zásadnější důvod, proč se nové tvůrčí přístupy ve výuce neuplatnily.

Důvodem je absence aprobovaných pedagogů zaměřených na ICT oblast, jejichž počet se podle výzkumů blíží až 70 %. Druhým důvodem, který byl zmíněn a prokázán v rámci výzkumu Katedry informačních technologií a technické výchovy Pedagogické fakulty UK je skutečnost, že většina dotázaných pedagogů považuje svoji erudovanost v oblasti informačních a komunikačních technologií za minimum, které je dostačující k výuce informatiky. Někteří dotázaní pedagogové deklarovali, že nedosahují ani této minimální úrovně. (Černochová 2013)

3.2.4 Strategie digitálního vzdělávání do roku 2020

> *„Informační technologie by měly prostupovat celým procesem výuky na základních školách, nikoli jen v předmětech typu ‚Práce s počítačem'. Plné zapojení moderních technologií do výuky všech předmětů vnímá stát jako nezbytné v rámci posunu vzdělávacího systému od prostého memorování faktů k důrazu na čtenářskou gramotnost, komunikační dovednosti a logické myšlení." (Strategie digitálního vzdělávání do roku 2020, s. 2).*

Vize digitálního vzdělávání by se dala, dle oficiálního dokumentu Strategie digitálního vzdělávání do roku 2020, shrnout do následujících bodů.

Otevřené vzdělávání: je nový přístup k modernímu způsobu vzdělávání. Mělo by se jím stát celoživotní vzdělávání, které nebude závislé na konkrétním místě, a tím lze příležitost nabídnout každému jedinci bez rozdílu. Systém otevřeného vzdělávání bude zajišťovat kvalitní přístup ke sdíleným zdrojům, jako např. levné a vysokorychlostní připojení nezávislé na lokaci nebo použité digitální technologie, či kvalitní vzdělávací obsah.

- **Digitální gramotnost:** jedná se o osvojení digitálních kompetencí, které sdružují vědomosti, dovednosti, postoje a hodnoty nezbytné pro tvořivé využívání digitálních technologií v široké oblasti lidského života.
- **Informatické myšlení:** je nový pojem, který popisuje způsob uvažování, který při řešení složitějších problému využívá informatické řešení, zahrnující schopnost analýzy, syntézy a zobecňování. Je zde rozvíjena schopnost přesné formulace problému.

- **Digitální technologie ve vzdělávání:** prostupují všemi oblastmi lidské činnosti, proto je nezbytné tento trend zohlednit i ve školním prostředí, kde se digitální technologie čím dál tím více zapojují do běžného vyučování, nejen informaticky zaměřených předmětů. Stávají se tak další pedagogickou a didaktickou pomůckou, díky které mohou učitelé ve svých hodinách nejen připravit různé formy netradiční výuky, ale také rozvíjet digitální gramotnost u svých žáků. Tento trend popisuje a také v budoucnu předvídá několik prestižních studií, z nichž k té nejprestižnější patří Horizont Report[2].

V závěrečném shrnutí této kapitoly jsou uvedeny hlavní cíle Strategie digitálního vzdělávání do roku 2020.

- **Snižování nerovností ve vzdělávání –** díky otevřenému vzdělávacímu systému bude poskytnuto vzdělávání všem, kdo o něj projeví zájem, aniž dojde k omezování vzhledem k socioekonomickému statusu, regionu nebo kulturně odlišnému prostředí. Aby ovšem mohl tento systém ve všech směrech splňovat své cíle, je nutné zajistit kvalitní přístup k těmto technologiím také mimo školu, což není možné v této době stoprocentně splnit.

- **Podpora kvalitní výuky a učitele –** z důvodů naplnění potřeb trhu práce, kdy stále více oborů je více či méně závislých na informačních dovednostech, je potřeba přehodnotit současné vzdělávací cíle, aby lépe reflektovaly budoucí požadavky na zaměstnance. Jedním z těchto cílů je celoživotní vzdělávání, které vychází ze stále častější potřeby rekvalifikace pracovních sil v důsledku mnohem kratší doby strávené v konkrétním zaměstnání. Aby se mohl tento trend plně zapojit do výuky, je třeba zvýšit pedagogické dovednosti v oblasti digitálních technologií. Jde především o informatické myšlení, které mohou poté pedagogové předávat svým žákům. Také jde o dovednosti uživatelské, potřebné např. pro tvorbu digitálních materiálů, což řada pedagogů prokázala tvorbou rozsáhlé databáze digitálních učebních materiálů v rámci projektu *EU peníze školám.* (Strategie digitálního vzdělávání do roku 2020, s.15)

[2] NMC (2014). THE NMC HORIZON REPORT: 2014 K-12 EDITION. [online] http://bit.ly/ZvclAm

3.2.5 RVP v oblasti informatiky a ICT

Jak již prokázalo několik výkumů, o kterých bylo pojednáno v předchozích kapitolách, byla oblast informatiky a ICT v rámci RVP dlouho opomíjenou, což mělo za následek, že rámcový vzdělávací program pro základní vzdělání (RVP ZV) nedoznal během 14 let výraznějších změn.

V důsledku plnění Strategie digitálního vzdělání do roku 2020 došlo k revizi tohoto závazného vzdělávacího dokumentu, který vymezuje kompetence a výstupy pro žáky základních škol. Tato revize je založena na celkové změně pojetí role informatiky v rámci celého průběhu vzdělávání základního školství, nejen v čistě informaticky zaměřeném předmětu. V následující podkapitole budou tyto změny blíže specifikovány.

Základní východiska a teze revizí ICT[3]

Změna rámcového programu je založena na šesti opěrných pilířích.

1. **Rozsah revize RVP ZV** – vzhledem ke změně pojetí informatiky je třeba část kompetencí, které zatím žáci získávali v rámci izolovaného předmětu, začlenit do výuky neinformatických předmětů, což bude mít za následek změny nejen v oblasti Informačních a komunikačních technologií, ale také v dalších oblastech.
2. **Časová dotace v učebním plánu** – rozvoj digitální gramotnosti a informatického myšlení je nutno rozvíjet v průběhu celé školní docházky, nejen nárazově v několika ročnících základní školy, jak je tomu doposud.
3. **Rozvoj digitální gramotnosti** – vzhledem k nízké časové dotaci izolovaného předmětu informatického charakteru, je nutné její rozvoj rozšířit na vhodné aktivity v různých vyučovacích předmětech

[3] Základní východiska a teze revizí ICT kurikula, Národní ústav pro vzdělávání [online]. [cit. 2019-07-06]. Dostupné z: http://www.nuv.cz/t/1-zakladni-vychodiska-a-teze-revizi-ict-kurikula

4. **Rozvoj informatických kompetencí** – informatické kompetence jsou stále častěji vyžadovány v mnoha profesích, nejen informatického zaměření, kde napomáhají řešení každodenních problémů a situací. Tento rozvoj je třeba začlenit do závazného vzdělávacího plánu již od počátku základního vzdělávání.

5. **Rozvoj oborových kompetencí dalších vzdělávacích oblastí** – vzhledem k rozšiřujícímu se vlivu ICT v nejrůznějších oblastech lidských činností vyžaduje změnu vzdělávací oblast RVP a její vzdělávací obsah.

6. **Využití digitálních technologií ve výuce** – začlenění digitálních technologií do vyučovacího procesu je nezbytně nutné pro rozvoj digitální gramotnosti žáků. Žák této schopnosti využívá při učení a vzdělávání se, což posléze vede k rozvoji celoživotního sebevzdělávání.

3.2.6 Projekt PRIM[4]

Strategický tříletý[5] projekt *Podpora rozvíjení informatického myšlení* (dále jen PRIM) vznikl díky poptávce po nové orientaci informatiky, směřující od uživatelského ovládání ICT k základům informatiky. Tento projekt je primárně zaměřen na podporu práce učitele. Projekt PRIM má za cíl rozvoj informatického myšlení v předmětu informatika na všech stupních škol od mateřských po střední.

Garantem projektu je Pedagogická fakulta Jihočeské univerzity v Českých Budějovicích. Participují na něm všechny pedagogické fakulty v ČR a Národní ústav ve vzdělávání.

Hlavními cíli projektu[6] je:

- vytvořit komplex vzájemně provázaných vzdělávacích materiálů (především metodických materiálů pro učitele) pro všechny věkové skupiny, tvorba vzdělávacích materiálů pro školy,

[4] Informatické myšlení [online]. 2017 [cit. 2019-07-04]. Dostupné z: https://www.imysleni.cz/
[5] Projekt probíhá od října 2017 do září 2020
[6] Informatické myšlení: O Projektu [online]. 2017 [cit. 2019-07-04]. Dostupné z: https://www.imysleni.cz/o-projektu

- vybudovat systém školení učitelů předmětu „Informatika“ pro všechny stupně škol,
- zpracovat jednotný systém přípravy budoucích učitelů na všech pedagogických fakultách, který zahrnuje předměty oborové a didaktické přípravy učitelů,
- oslovit veřejnost popularizačními kampaněmi, které by měly upoutat pozornost a naladit společnost příznivěji k IT povoláním a významně ovlivnit změnu negativního postoje společnosti k technickým povoláním.

3.3 Programovací jazyky

Pro výuku programování a algoritmizace je možné zvolit různé programovací jazyky, kdy vhodná volba programovacího jazyka může usnadnit prvotní seznámení s danou problematikou a zvýšit zájem o další vzdělávání v této oblasti. Naopak nevhodná počáteční volba programovacího jazyka může žáky od programování předčasně odradit.

Následující kapitola představuje tzv. propedeutické jazyky, které jsou pro seznámení se světem algoritmizace nejvhodnější, a to na základě několika kritérií, které jsou podrobněji rozebrány. Pro ucelenost informací jsou v druhé části kapitoly zmíněny také klasické programovací jazyky, které vzhledem ke svému pojetí nejsou pro začátečníky vhodné.

Při rozhovoru dvou lidí je základním verbálním komunikačním nástrojem společný jazyk, díky němuž se tito dva lidé mezi sebou dorozumí. Stejná znalost komunikačního nástroje je nutná v případě „rozhovoru“ uživatele s počítačem. Komunikačním nástrojem se v tomto případě stává tzv. programovací jazyk, který je v následujících kapitolách členěn do různých skupin a podskupin.

Existuje mnoho způsobů, jak definovat programovací jazyk.

> *„Je to umělý jazyk určený pro zápis počítačového programu, ve kterém jsou přesně definované syntaxe a sémantika.“ (Vitovský 2006)*

> *„Programovací jazyk je prostředník mezi běžnou řečí a posloupností typicky binárních číslic.“ (Kolář 2006)*

> *„Programovací jazyk je konečná množina příkazů (převáděných do binární formy), která má specifickou syntaktickou strukturu a pevně a přesně vymezenou sémantiku.“ (Kolář 2006)*

Jednotlivé programovací jazyky se od sebe liší svou syntaxí a sémantikou, které mohou mít velmi podobnou formu jako je tomu například u cizích jazyků.

Müller (2002) charakterizuje syntaxi programovacího jazyka jako souhrn pravidel udávajících přípustné tvary dílčích konstrukcí a celého programu. Sémantika v obecném významu pojednává o významu slov a znaků. Nejinak je tomu v případě programování, kde je sémantikou přiřazen význam dílčích programových konstrukcí, které splňují syntaktické požadavky.

3.3.1 Propedeutické programovací jazyky

Pokud člověk k programování používá programovací jazyk, téměř vždy narazí na stejné úskalí. K vytvoření prvního programu, byť pouze s textovým výstupem, je nutné napsat několik řádků programového kódu. Aby bylo možné něco takového vytvořit, je potřeba zvládnout základní syntaxe a příslušné sémantiky, a s tím pomáhají propedeutické programovací jazyky. Pro správné pochopení významu propedeutických jazyků je vhodné vysvětlit samotný termín propedeutika.

> *„Průprava. Všeobecná příprava ke studiu určitého oboru, často ve formě úvodního kurzu na vysokých školách apod.“ (Průcha 2013, s. 185)*

Slovo propedeutika lze rozdělit na dvě části. První částí je slovo *pro*, které lze v řečtině přeložit slovem *před,* druhou část představuje slovo *pedeutika*, jež vychází z řeckého slova *paideúein* a lze ho přeložit jako *vyučování*. Z toho lze odvodit, že slovo *propedeutika* označuje předběžnou výuku či úvod do určité vědy. Vzhledem k tomu, že předmětem této propedeutiky je programování, jedná se o programovací jazyk, který příznivě rozvíjí předpoklady pro následné studium dalších programovacích jazyků. (Propedeutika 2017)

Při tvorbě programového kódu mohou žáci využívat různé editory podporující konkrétní programovací jazyk, což velmi usnadňuje zápis programového kódu. Přes veškerá

zjednodušení, spočívající v dokončování názvů příkazů, hlídání syntaxe a zvýrazňování různých částí programového kódu, jim neodpadá znalost jednotlivých příkazů nebo alespoň jejich začátků.

Předností propedeutických programovacích jazyků je především velmi rychlá odezva na programový kód, která je na rozdíl klasických jazyků v grafické podobě, což je pro děti a žáky mnohem atraktivnější než čistě textový výstup.

Neméně důležitou skutečností je lokalizace většiny propedeutických jazyků do českého jazyka, čímž odpadá problematika s překládáním. Díky tomu jsou tyto jazyky přístupné široké skupině dětí různého věku, kdy jediným omezením je schopnost ovládat dané prostředí a orientace v něm.

Propedeutické programovací jazyky jsou primárně zaměřeny na rozvoj algoritmického myšlení. Žáci se nemusí zabývat způsobem zápisu, ale místo toho se mohou cele soustředit na jednotlivé části programu a jejich logickou návaznost.

Klíčový proces, nutný pro úspěšné vytváření počítačového programu, je analýza problému, jež vede k rozdělení problému na elementárnější části. Následuje řešení dílčích problémů na základě různých programových struktur, tedy algoritmické myšlení. Konkrétní typy propedeutických programovacích jazyků budou zmíněny v následujících podkapitolách.

Cílem není vychovat z každého žáka programátora, ale naučit žáky algoritmicky přemýšlet. Proto by se tyto jazyky měly primárně využívat na základní škole, případně v předškolní výuce.

Členění propedeutických programovacích jazyků je závislé na míře využití informačních technologií. K výuce algoritmizace není počítač zcela nezbytný, jak bude popsáno v další části bakalářské práce.

Srovnání propedeutických programovacích jazyků

Kapitola je zaměřena na vzájemné srovnání propedeutických programovacích jazyků. Pro porovnání jsou zvolena srovnávací kritéria, která mohou pomoci při volbě vhodného

propedeutického jazyku. Těmito kritérii jsou věková cílová skupina, náročnost nasazení daného prostředí do výuky a způsob využití při rozvoji algoritmického myšlení.

Jedním z hlavních parametrů, které rozhodují o nasazení daného programu ve škole, je její licenční politika. Vzhledem k rozpočtu školy jsou zvoleny převážně volně dostupné produkty, které jsou zcela zdarma, případně mají zkušební verzi s určitým omezením bránícím plnému nasazení do výuky. Pokud je do výčtu zařazen placený software, je to z důvodu velké rozšiřitelnosti mezi učiteli informatiky. Většina licencovaných produktů má vytvořen velmi dobrý multilicenční systém zaměřený právě na oblast vzdělávání, tudíž nic nebrání jejich využití při hodinách ve škole.

Při výběru vhodné metody je třeba mít na paměti několik důležitých kritérií. Metodu je třeba přizpůsobit věku žáků. Pro předškoláky a mladší žáky je nutná atraktivita zvyšující motivaci k rozvíjení algoritmického myšlení. K tomu napomáhá grafické zpracování a systém odměňování při úspěšném plnění zadaných úkolů. Nejvhodnější formou těchto zástupců jsou propedeutické jazyky, využívající herního prostředí, které přirozenou a nenásilnou formou provází žáky celým procesem učení. Neméně důležitý je samotný výstup z vytvořeného programovacího kódu, který by měl být také vzhledově zajímavý. Dalším hlediskem je množství osvojených příkazů potřebných k tvorbě vybraného programu. Vhodné jsou programovací prostředí na způsob tutoriálů, ve kterých se množství dostupných příkazů postupně rozšiřuje, aby nedošlo k počátečnímu zahlcení žáků nadbytečnými složitostmi. Tato funkce je vhodná pro samostudium, ve výuce je nahrazena metodickým vedením pedagoga.

Z výše uvedeného textu vyplývá, že pro předškolní děti a mladší žáky jsou vhodnější programovací jazyky, ve kterých je zápis programového kódu nahrazen sadou ikon, případně celými bloky. Každý blok představuje soubor příkazů nebo programovacích konstrukcí. Typickými příklady těchto programovacích jazyků jsou jazyky stavebnicové, případně ikonické.

Posledním důležitým hlediskem, které napomáhá ke snadnějšímu zvládnutí této dovednosti, je české prostředí, a to nejen při samotné stavbě programu, ale také v dostupné nápovědě.

Pokud vybíráme metody pro žáky, kteří mají výše zmiňované základy osvojené či budou při své práci využívat pokročilejších funkcí klasických programovacích jazyků, je nutné se zaměřit na funkcionální možnosti dané metody.

Při volbě vhodného programového prostředí je třeba mít na paměti různá funkcionální omezení, která jsou závislá na využití daného programu.

Toto omezení souvisí především s variabilitou a složitostí programového kódu, který lze v daném prostředí vytvářet. Některá programová prostředí jsou úzce zaměřena na konkrétní činnost. Dají se v nich vytvářet pouze programy obdobného charakteru, které jsou založeny na stejném principu. V případě, že by je uživatel chtěl použít i k jiným činnostem, musel by přistoupit k nestandartnímu řešení.

Mezi omezení patří absence tvorby vlastních proměnných, což jsou zcela běžné a hojně využívané konstrukční prvky v algoritmizaci. Na druhou stranu uvedená omezení pociťuje pouze uživatel, který má již zkušenosti s jinými programovacími jazyky. Pedagog by měl vždy zvolit pracovní zadání, ve kterých žáci tyto nedostatky nepocítí.

Příkladem tohoto prostředí je KODU Game Lab, které je v kap. 4.1.6 podrobněji popsáno.

Programování bez počítače

Následující kapitola pojednává o počátcích programování, které jsou charakteristické svou nezávislostí na výpočetní technice a počítačovém prostředí. Jedná se o specifický přístup k rozvoji algoritmického myšlení. Programování bez počítače lze využít ve vyučovacích hodinách na 1. stupni a při mimoškolních aktivitách. Při používání této metody není nutné využívat počítače, dá se rozvíjet i v jiných předmětech, než je výuka informatiky, například v tělocviku, výtvarné výchově, a v neposlední řadě i v matematice.

Programování bez počítače, tzv. unplugged metody, se rozdělují do čtyř kategorií dle objektu programování:

- programování spolužáka,
- programování pomocí symbolů,
- řazení známého postupu,
- programování robota.

Jednou z aktivit rozvíjející algoritmické myšlení, je hra na živého robota. Metoda byla ověřena autorem práce v rámci zájmového kroužku Programování na Základní škole Karla IV. v Ústí nad Labem. Aktivita spočívá v ovládání jiného spolužáka pomocí předem dohodnutých příkazů. Tyto příkazy mohou být verbální nebo nonverbální. V případě, že je k vedení spolužáka využito verbálních příkazů, je dobré žákovi, představujícímu robota, zavázat oči, aby byla eliminována možnost ovlivňovat průběh procesu. Na začátku je třeba vytyčit trasu, kterou musí žák - „robot“ projít, aniž by narazil do překážek, které se na ní vyskytnou. Úspěšnost žáka – „robota“ závisí na přesnosti pokynů žáka – „programátora“. Hru na živého robota lze hrát prakticky kdekoli, ve třídě se dá vytvořit bludiště například ze školních lavic. V případě realizace hry v přírodě je možné využít nejrůznější přírodní překážky, a záleží jen na fantazii tvůrce, jak náročnou trasu vytvoří.

Žák - „programátor“ má několik možností, jak při tvorbě příkazů postupovat. „Programátor“ si projde vytyčenou trasu sám, zaznamená sled svých pohybů, a na základě vlastní zkušenosti provede žáka - „robota“, aniž by došlo ke kolizi s překážkami na trati. Obtížnější varianta spočívá ve vytvoření sledu příkazů, bez toho, aby si „programátor“ prošel trasu. Veškeré příkazy jsou tvořeny jen na základě vizuálního tvaru trasy.

Hra na robota se může hrát i s předškolními dětmi, kteří již mají povědomí o časové souslednosti událostí a jsou schopné je správně uspořádat. Hlavním cílem hry na robota je tvorba ovládacího programu, složeného z instrukcí v určitém pořadí, které se nemusí specificky zapisovat. Podmínkou ke zdárné realizaci je domluva „programátora“ a „robota“ na instrukcích, které je třeba dodržovat. Jednou z možností je domluva na určitých zvukových signálech, které reprezentují změny v pohybu robota.

Obdobou hry na robota je grafické programování, při kterém žáci ovládají virtuální objekt pomocí předem zadané sady příkazů. Tento způsob vyučování vede k rozvoji lineárního programování, při kterém žáci celý program zapisují formou jedné řady příkazů, případně vytváří pro opakující se úkony sekvenci příkazů. Sekvenci mohou posléze označit jako funkci či proceduru, díky níž lze zápis celého programu tvořit mnohem efektivněji a přehledněji. Výše zmiňovaný přístup vede žáky k úspornějšímu zápisu programu, který je v dnešní době trendem moderního programování. Metoda je podrobněji popsána v podkapitole *Grafické programování na papíře*.

Další možností, jak žákům přiblížit algoritmizaci bez počítačů, je kartičková metoda. Hlavním cílem žáků je poskládat karty s předem připravenými událostmi do správného pořadí tak, aby se dosáhlo zadaného cíle, například žáci musí seřadit činnosti konající se při cestě do školy. Těžší variantou je popis pracovního postupu, který je žákům všeobecně znám, například vaření čaje. Popis a uvědomění si jednotlivých elementárních úkonů dané aktivity může být pro žáky obtížné, bude zde nejspíše zapotřebí pedagogovy pomoci.

Programování s počítačem

K výuce algoritmizace a programování lze přistupovat mnoha způsoby. K výuce různých algoritmických postupů a k jejich osvojení je možné zvolit několik různých metod. Tato kapitola se zabývá srovnáním jednotlivých metod a představením zástupce, kterého lze v této metodě využít.

Textově orientované propedeutické programovací jazyky jsou jedním ze základních typů programovacích jazyků, a jejich interpretace je založena na zápisu samotného programového kódu, což sebou nese jisté úskalí, kterým je syntaxe daného kódu. Žák musí znát nejen příslušný příkaz kódu, ale musí znát základy slovní zásoby daného programovacího jazyku a parametry, které jsou pro správnou funkci vyžadovány. Pro zjednodušení práce v daném programovacím prostředí existuje velmi často výčet přípustných příkazů společně s vysvětlením funkce, čímž odpadá nutnost se tyto příkazy učit zpaměti, jak je to často nutné u klasického programování. Názvy jednotlivých příkazů jsou intuitivní, vycházejí z výsledného efektu daného příkazu a žák si je dokáže rychle osvojit již při krátkodobém používání.

Při programování je nezbytné veškeré příkazy zapisovat ručně, což není zcela vhodné pro děti nižších věkových kategorií. Výhodou tohoto zápisu je možnost libovolného rozšíření portfolia na základě stávajících příkazů, čímž je možné si daný jazyk upravit dle individuálních požadavků. Při vymýšlení nových příkazů si žáci osvojí efektivnější zápis opakujícího se programového kódu, čímž se program stává přehlednějším. Dále jsou schopni rozebrat komplexní úlohy na menší samostatné celky, což je základní myšlenkou algoritmizace. Získané dovednosti mohou později využít při tvorbě procedur a funkcí u klasických programovacích jazyků.

Základní programovací konstrukcí těchto jazyků je cyklus definovaný počtem opakování, který může sdružovat frontu více příkazů tvořících logický celek. Další konstrukcí je tzv. *rekurze*, kdy samotný příkaz je volán z těla daného příkazu. Rekurzivní způsob programování je převážně v počátcích pro žáky velmi náročný, jelikož narozdíl od lineárního programování představuje abstraktní posloupnost kroků, kteru si žáci hůře představují. Hlavními představiteli je programovací jazyk Logo (viz kap. 4.1.1), Karel (viz kap. 4.1.2) a Petr.

Druhou skupinou jsou stavebnicově orientované jazyky, jejichž hlavní předností je absence zápisu jednotlivých příkazů a programovacích struktur. Díky této koncepci odpadá problém s nesprávnou syntaxí, kterou je nutno dodržovat u předchozí skupiny programovacích jazyků. Stavebnicově orientované jazyky se vyznačují typickým uspořádáním, které připomíná skládání dílků puzzle. Do jisté míry je tím eliminována možnost vzniku zásadní chyby, protože se spojují pouze prvky určitého typu, které mají logickou souvislost a návaznost. Jak již bylo zmíněno, nedochází k chybám v syntaxi, protože žák příslušný příkaz nezapisuje, ale vybírá si z připravených „příkazových“ modulů. Liší se tím od ikonických jazyků, kde je každému příkazu přiřazena ikona určitého vzhledu. Mezi stavebnicově orientované jazyky patří programovací nástroj Scratch, který bude ještě podrobněji zmíněn.

Třetí skupinou jsou kartičkově orientované programovací jazyky, které jsou postaveny na obdobném principu jako jazyky stavebnicové. Také u tohoto typu programovacího jazyka není třeba znalost syntaxe programového kódu, jelikož je nahrazen grafickou interpretací. Zásadním rozdílem mezi stavebnicovým a kartičkovým programovacím jazykem je

skladba programu, která se odlišuje možností libovolného umístění jakékoliv z připravených karet do programového kódu, čímž není možné zajistit stejnou kontrolu jako v předchozím příkladu. To dává žákům větší volnost při vytváření programu a více prostoru pro tvorbu chyb, které následně musí odstraňovat, a tím také více přemýšlet o celém programu i jeho částech.

Nevýhodou těchto programovacích jazyků je omezená lokalizace, kde dominantní úlohu hraje anglický jazyk. Z toho důvodu nejsou u nás tyto jazyky tolik rozšířené a známé, ačkoli jsou svým zaměřením určeny pro využití na základní škole. Mezi typického představitele patří programovací prostředí Etoys.

Ikonické programovací jazyky sázejí na jednodušší tvorbu programového kódu, kdy se žák může plně soustředit na algoritmickou část úlohy, aniž by se coby „programátor“ zatěžoval syntaktickou stránkou daného problému.

Programový kód je vytvářen v programovacím prostředí grafického charakteru, které je možno ovládat pouze myší, případě herním ovladačem. Žák má na výběr z nabídky ikon, z nichž každá zastupuje určitou funkci nebo příkaz. Ikony jsou skládány do fronty příkazů, čímž se vytváří program. Typickým představitelem tohoto typu jazyku je programovací jazyk Baltík od firmy SGP Systém, který se stal jedním z nejrozšířenějších jazyků na základních školách.

3.3.2 Klasické programovací jazyky

V této kapitole bude stručně pojednáno o klasických programovacích jazycích. Klasické programovací jazyky sice nejsou předmětem této práce, ale jsou nedílnou součástí programovacích dovedností, a je zde také integrace zápisu syntaxe do mnoha propedeutických jazyků.

Hlavním rozdílem mezi propedeutickým a klasickým programovacím jazykem je možnost využití složitějších datových struktur a algoritmických konstrukcí, které jsou při tvorbě náročnějšího programu nezbytné pro jeho funkci. Jelikož je však tato tvorba spojena se zápisem těchto konstrukcí v příslušném programovacím jazyku, je jejich zařazení vhodnější k výuce na střední škole.

Důvodů pro zařazení klasických programovacích jazyků na střední, případně vysoké školy, je několik. Prvním z nich je znalost syntaxe příslušného programovacího jazyka a druhým je určitá neatraktivnost, který spočívá v textovém nebo jednoduchém grafickém výstupu.

Jak bylo již zmíněno, propedeutické jazyky slouží primárně k osvojení základních programovacích dovedností, kam je v prvé řadě řazeno algoritmické myšlení a osvojení základních programovacích konstrukcí, které žáci později využijí při výuce klasického programovacího jazyka. Zde je dobré zdůraznit, že správná volba prvotního jazyka má velký vliv na pochopení následujících programovacích struktur.

Vzhledem k velkému důrazu na objektově orientovaný přístup k programování je výhodné volit takové zástupce propedeutických jazyků, které jsou ve své podstatě založeny na stejném přístupu. Typickými zástupci jsou jazyky, které jsou využívány v programovém prostředí Kodu Game Lab a Scratch. Tyto programy jsou však závislé na výpočetní technice, která v nižších ročnících základní školy nebo předškolní výuce nemusí být vždy dostupná, respektive dostupná v dostatečném počtu. U dětí těchto věkových kategorií je vhodné využívat programovací roboty, jejichž ovládání a řízení probíhá přímo v přístroji, případně skrze čidla, reagující na změnu prostředí. Při této práci si žáci nejlépe osvojí tvorbu systematických kroků vedoucích k zadanému cíli.

Jiné jazyky mohou zobrazovat vytvořený programový kód pomocí ikonického jazyka nebo soudobého programovacího jazyka, typicky C#. Při použití těchto jazyků je poté mnohem snadnější cesta k přechodu k samotnému C# nebo jiným programovacím jazykům.

Ať už je pro začátky ve výuce programování a algoritmizace zvolen jakýkoliv typ, nesmí se zapomínat na skutečnost, že každý z uvedených propedeutických jazyků má své typické zaměření, z čehož plyne, že nelze od těchto jazyků požadovat stejné možnosti jako od programovacích jazyků v klasickém programování.

3.4 Robotické programovatelné hračky

Tato kapitola je věnována novému trendu výuky algoritmizace, která je pro žáky mnohem atraktivnější a zábavnější. Jedná se o stavebnice a robotické hračky s možností změny jejich chování díky vlastnímu programovému kódu. Žákům se dostává okamžité odezvy na jimi vytvořený program, což zvyšuje jejich motivaci. V následujících kapitolách jsou zmíněni nejrozšířenější zástupci, kteří se používali v minulosti, nebo jsou dostupní na současném trhu.

Programovatelné robotické hračky jsou ideální pro využití v začátcích programování, jelikož žáci své vytvořené programy rovnou převádějí do mechanické hračky, která vykonává jimi vytvořenou sekvenci příkazů. Tyto robotické hračky nebo stavebnice jsou díky této funkci pro žáky velmi atraktivní, je však třeba dát velký pozor na to, zda jejich nadšení plyne z novosti této pomůcky nebo zda za tímto nadšením stojí jiný přístup při tvorbě programu. Velkou nevýhodou, zvláště ve školním prostředí, je jejich finanční náročnost. Stavebnice je vhodné zakoupit ve větším množství, a proto k pořízení těchto pomůcek je ve většině případů využita některá z dotačních výzev. (AV News. 2018(3).)

4 Ukázka vybraných propedeutických nástrojů při výuce algoritmizace a programování

Jak již bylo několikrát zmíněno, pro rozvoj algoritmizace a programování u dětí jsou nejvhodnější propedeutické jazyky a prostředí. Přestože se v práci často vyskytuje termín propedutický jazyk, ve skutečnosti existuje pouze málo zástupců těchto jazyků. V převážné většině jde o tzv. propedeutické programovací prostředí, které je uzpůsobeno tak, aby žáci nebyli zahlceni zbytečnými informacemi a díky tomu bylo pro ně zvládání jednotlivých úkolů snadnější. V následující podkapitole jsou vybraní zástupci těchto jazyků a prostředí blíže specifikovány. Toto rozdělení je z hlediska využití v pedagogické praxi nepodstatné, jelikož oba přístupy sledují stejný cíl.

4.1 Vybraní zástupci propedeutických programovacích jazyků

Jak už bylo popsáno výše, k rozvoji algoritmizace a programování lze využít nemalé množství propedeutických programovacích jazyků. Přestože tyto jazyky v zásadě sledují stejný cíl, k jejich dosažení jsou používány rozličné metody. Propedeutické jazyky jsou porovnávány na základě srovnávacích kritérií, k nimž patří cílová věková skupina, znalost cizího jazyka, nutnost porozumění textu, objektový nebo neobjektový přístup, a v neposlední řadě pořizovací cena. Předchozí kapitoly pojednávaly o hlavních rozdílech propedeutických programovacích jazyků, popisovaly klady a zápory jednotlivých typů a metod. Výběr těchto jazyků probíhal na základě osobní zkušenosti autora, komunikace s ostatními učiteli informatiky či historického významu, na jejichž základě později vznikly další varianty a modifikace tohoto programovacího jazyka. Z důvodu ucelenosti mohou některé skupiny jazyků obsahovat více než jednoho představitele.

4.1.1 Logo a jeho varianty

Jazyk Logo je jednoduchý funkcionální programovací jazyk, který od svého vzniku v roce 1967 prošel mnoha obměnami[7] a k letošnímu roku existuje 197 implementací tohoto jazyka[8].

Hlavním představitelem tohoto jazyka je želva, která svým pohybem po pracovní ploše, či papíru, zakresluje trajektorii daného pohybu. Tímto způsobem je vytvářena tzv. *želví grafika*.

Myšlenka Loga je velmi podobná skutečné želvě, která se pohybuje v písku a svým ocáskem kreslí obrazec. Původně jazyk Logo ovládal skutečného robota, v pozdějších letech se celá koncepce převedla jen do virtuálního prostředí.

Pro účely školního prostředí jsou vhodné dvě varianty, přeložené do českého jazyka. První implementací je Imagine Logo, které je zatíženo licenční smlouvou,[9] a proto by mělo být zařazení tohoto programu do výuky dobře zváženo.

Prodává se ve třech licencích:

- Neomezená školní licence, kterou lze využít pro všechny školní počítače, a zároveň jí mohou učitelé a žáci použít na libovolném počtu počítačů mimo školu.
- Omezená školní licence, která se od neomezené licence liší maximálním použitím na dvaceti školních počítačích.
- Domácí licence, kterou lze využít na libovolném počtu domácích počítačů.

Přestože se jedná o kompletně objektově orientovaný jazyk, jehož koncepce je základní myšlenkou všech moderních programovacích jazyků, lze pro první seznámení

[7]Logo (programovací jazyk). In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2017-07-07]. Dostupné z: https://cs.wikipedia.org/wiki/Logo_(programovac%C3%AD_jazyk)

[8]LOGO. In: *Kids, Code, and Computer Science: Helps kids code + explore computer science* [online]. 2014 [cit. 2017-07-07]. Dostupné z: https://www.kidscodecs.com/logo-programming-language/

[9]Jak získat Imagine. *Imagine* [online]. 2002 [cit. 2017-07-07]. Dostupné z: http://imagine.input.sk/cz/tutorial.html

s programováním využít neobjektového přístupu. Výhodou je možnost přechodu na objektově orientované programování v pozdějším věku.

Vzhledem k nutnosti zápisu jednotlivých sekvencí programovacího kódu není tento program vhodný pro předškolní děti, či pro žáky prvního stupně ZŠ.

Imagine Logo je nepřímým nástupcem druhé instance orientované na školní prostředí, kterou je Comenius Logo. Tato varianta je již zastaralá, a přestože byla vydána ve volně šiřitelné licenci, pro dnešní moderní operační systémy je nepoužitelná. Navzdory těmto faktům lze najít celou řadu programovacích prostředí, která jsou na tomto programovacím jazyku postavena. Tímto prostředím může být např. EasyLogo[10], což je volně šiřitelná desktopová aplikace, která žáky naučí základnímu programátorskému myšlení na základě tvorby cyklů a rekurze.

V tomto programovacím jazyku se příkazy zapisují na základě ikonické interpretace, díky nimž není třeba znalost konkrétních příkazů, což umožňuje využití v předškolním vzdělávání nebo v nižších ročnících ZŠ. Žáci se věnují pouze tvorbě algoritmů a nezatěžují se jejich zápisem. Vzhledem k postupnému zvyšování obtížnosti a rozšiřování dostupných nástrojů, lze tento jazyk použít k samostudiu algoritmizace. Napomáhá tomu průvodce, který uživatele na začátku seznámí se základy, které jsou potřebné pro další úrovně.

Autor práce tuto skutečnost ověřil v rámci hodin informatiky v 5. ročníku ZŠ, kdy žáci i bez předchozích znalostí byli v převážné většině schopni projít všechny počáteční úrovně. Obtíže nastaly v okamžiku zapojení cyklů, jelikož si žáci nedokázali představit správný sled opakujících se událostí. Přesto byli někteří žáci, pomocí metody pokus a omyl, schopni i tyto úrovně splnit.

[10]EasyLogo [online]. [cit. 2019-07-05]. Dostupné z: http://edi.fmph.uniba.sk/~salanci/EasyLogo/index.html

4.1.2 Karel

Stejně jako programovací prostředí jazyka Logo, byl v Čechách v 80. letech 20. století nejrozšířenějším programovacím prostředím jazyk Karel. Přestože se jedná o starší programovací prostředí, v sekvenčním programováním hraje nezastupitelnou roli. Hlavní metodou programovacího jazyka Karel je tzv. rekurze, která je velmi rozšířena v klasických programovacích jazycích.

Programovací jazyk byl vyvinut na přelomu 70. a 80. let 20. století učitelem programování ze Stanfordské univerzity Richardem E. Pattisonem[11], který ho pojmenoval na počest Karla Čapka, autora divadelní hry RUR – Rossums Universal Robots. V původním podání, které se k nám dostalo díky zásluze Doc. Jozefa Hvoreckého, Csc., měl program mnohem více příkazů, než v jakém podání ho známe v České republice. Díky tomuto zjednodušení, o které se největší měrou přičinili Ing. Tomáš Bartovský, Csc. a Ing. Rudolf Pecinovský, Csc., se jazyk Karel mohl více přiblížit dětem. Oproti původnímu návrhu byla česká verze rozšířena o tzv. rekurzi, při níž je určitá procedura nebo funkce znovu volána dříve, než je dokončeno její předchozí volání. (Rekurze (programování). In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 4. 12. 2017 [cit. 2019-07-05]. Dostupné z: https://cs.wikipedia.org/wiki/Rekurze_(programov%C3%A1n%C3%AD))

Karel je robot, který se díky základní sadě příkazů, kterou lze rozšířit o vlastní programové sekvence, pohybuje po obdelníkové šachovnicové síti nazvané dvorkem pouze ve svislém a vodorovném směru. Kromě pohybu po těchto polích může robot pokládat či zvedat tzv. značky.

Stejně jako v případě výše popisovaného jazyku Imagine Logo, je třeba zapisovat programovací kód ručně, což znamená, že je vhodnější pro starší žáky ZŠ. Také rekurzivní opakovaní je pro žáky nižších ročníků nesnadno představitelné.

[11] Autor prvního programu a knížky „Karel The Robot“

Výhodou tohoto jazyka je volná licence či online verze, která umožňuje její bezplatné využití ve školním prostředí. Nutno však dodat, že vzhledem k minimalistickému vzhledu není pro dnešní žáky příliš atraktivní, a proto byl do výčtu zařazen spíše z historických důvodů.

Alternativní variantou je objektivně orientovaná verze označená názvem Karel++ (po vzoru C++). Modifikací programovacího jazyka Karel++, založeného na programovacím jazyku C# je Karel H.

4.1.3 Scratch[12]

Scratch je navržen pro cílovou skupinu dětí od 9 do16 let, čímž se nabízí jeho zařazení mezi vzdělávací pomůcku pomáhající v rozvoji algoritmického myšlení a programování na základní škole. Tomuto zařazení napomáhá nejen volná licence, ale také online varianta, díky níž je možné ho využít kdekoliv. Díky tomu se programovací jazyk Scratch vyřazuje ze skupiny edukačních metod, nepočítáme-li úzce zaměřené systémy na vývoj počítačových her.

Jedna z prvních vět, kterou si lze na stránkách autorů přečíst, shrnuje veškeré náležitosti vhodné výukové metody.

> *„Scratch pomáhá mladým naučit se myslet tvořivě, přemýšlet systematicky a spolupracovat — podstatné dovednosti pro život v 21. století.“*[13]

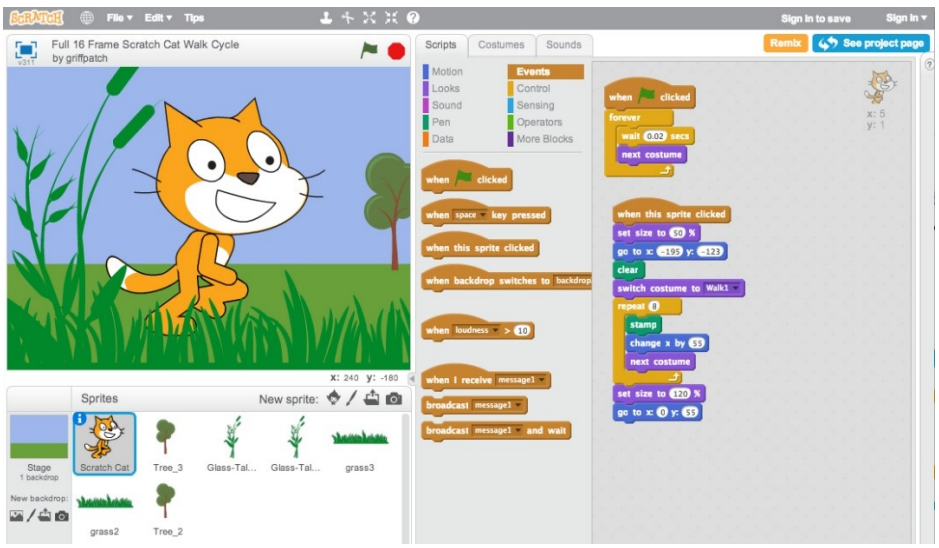
Toto tvrzení podtrhuje hned několik faktů. Lokalizace do více než padesáti světových jazyků, propracovaný systém tutoriálů, rozsáhlé nápovědy pro jednotlivé oblasti či konkrétní příkazy. Tutoriály mohou žáci využít při hledání postupů konkrétních algoritmů, aniž by museli čekat na pomoc učitele. Tím se z tohoto programovacího nástroje stává nejen vynikající pomůcka pro samostudium, ale také výborná didaktická pomůcka badatelsky orientovaného učení, které rozvíjí samostatné myšlení a učí řešit problémy. (Dostál 2015)

[12] Scratch - Imagine, Program, Share [online]. [cit. 2019-07-05]. Dostupné z: https://scratch.mit.edu/
[13] Scratch - O aplikaci. Scratch - Imagine, Program, Share [online]. [cit. 2019-07-05]. Dostupné z: https://scratch.mit.edu/about

Jak ve své práci uvádí L. Samková (2011, s. 336–341) badatelský proces lze chápat jako *„činnost, při které pozorujeme, dedukujeme, nabízíme hypotézy, snažíme se je ověřit, nemusíme dojít k žádnému konečnému závěru, závěry závisí na našem momentálním rozhledu a různí badatelé mohou interpretovat stejná fakta různě. Poslední tři znaky bádání v sobě skrývají onen most mezi teorií a praxí, mezi učebnicí a každodenní realitou. Jsou klíčem ke správnému chápání světa kolem nás.“*

Hlavním znakem této skupiny programovacích jazyků je využití předem připravených částí, ze kterých poté žáci skládají daný program.

Celé programovací prostředí, které se opět přesouvá pouze do 2 D prostoru, je rozděleno na oblast náhledu, zdrojového kódu a nabídky jednotlivých příkazů a konstrukcí, které jsou logicky členěny do jednotlivých kategorií, což usnadňuje jejich hledání. Velkou předností tohoto prostředí je možnost vytvářet prakticky veškeré programovací konstrukce, se kterými se žáci mohou setkat při programování pomocí libovolného klasického programovacího jazyka. Při tvorbě programu (hry) lze využít rozsáhlé galerie grafických objektů (většina z nich disponuje několika variantami téhož typu, což napomáhá snadnější animaci objektu), případně lze využít integrovaného grafického editoru nebo již vytvořených grafických předloh.



Obrázek 1 - Ukázka programového prostředí SCRATCH
zdroj: [online]. In: . [cit. 2019-07-07]. Dostupné z: https://aace-web-static.s3.amazonaws.com/2018/07/scratch.jpg

4.1.4 Etoys

Etoys lze zařadit mezi programová prostředí kartičkového typu. Prostředí lze rozdělit na dvě hlavní části, z níž jedna je tvořena grafickým editorem, ve kterém si žáci připraví potřebné rastrové grafické předlohy, a druhou je část programová. Programová část je méně intuitivní než v případě programovacího jazyka Scratch, navíc bez znalosti anglického jazyka je dosti nepřehledná a obtížná. Programová část se skládá z kompletní nabídky jednotlivých příkazů, které si musí žák poskládat dle svého vytvořeného algoritmu.

Tento programovací jazyk byl do této práce zařazen zejména kvůli úplnosti přehledu, přestože nelze předpokládat hromadnější zapojení do výuky, vzhledem k výše zmíněným důvodům. K atraktivitě nepřispívá ani vizuálně zastaralé programovací prostředí.

4.1.5 Baltík a jeho varianty

Mezi základní programovací jazyky patří programovací jazyk Baltík od společnosti SGP Systems, který je v České republice velmi rozšířen.

Přestože programovací jazyk Baltík patří mezi starší jazyky, je stále v mnoha základních školách velmi oblíben. První programové prostředí s označením Baltazar vzniklo v roce 1993. Výhodou tohoto systému je možnost přechodu do prostředí klasického programování, který nabízí programovací jazyk C, potažmo C# v případě *Baltie C# 3D*.

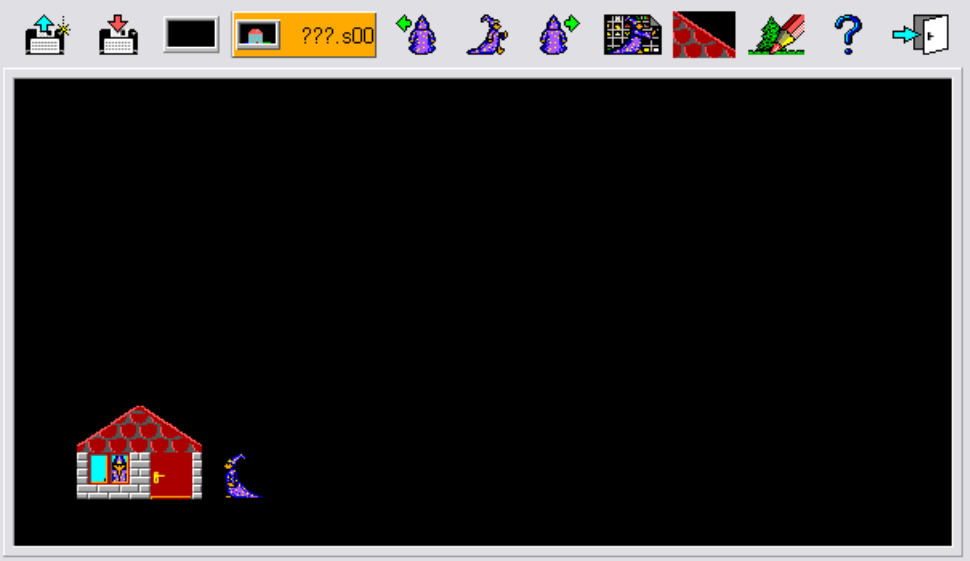
Celý systém je založen na 2D grafickém prostředí, které může být v dnešní době pro žáky neatraktivní a mít za následek jejich obtížnější motivaci. Kapitola je zaměřena na popis 2D verze Baltík 3.0, ve kterém autor vyučoval několik let programování v rámci zájmového kroužku.

Celý systém je rozdělen do 3 úrovní obtížnosti[14]

Skládej scénu – zde se mladší děti naučí pomocí počítačové myši skládat z připravených dílků (ikon) větší obrázky, které se v Baltíkovi nazývají scéna.

[14]SGP Systems - Baltík C# 3D vizuální výukové programovací nástroje pro děti, mládež a dospělé [online]. [cit. 2019-07-05]. Dostupné z: https://www.sgpsys.com/cz

Čaruj scénu – je první úroveň programového prostředí, ve kterém se žáci učí čaroděje ovládat. Tato dovednost je zatím omezena pouze na pohyb v ploše a čarování příslušných dílků, ze kterých mohou skládat větší celky. Tato úroveň je velmi podobná úrovni předchozí, pouze s tím rozdílem, že výsledný obraz vytváří čaroděj díky svému kouzlení, a ne žák kurzorem myši. Ovládací panel na této úrovni je velmi omezen.



Obrázek 2 - Baltík - režim Čaruj scénu
zdroj: [online]. In: . [cit. 2019-07-07]. Dostupné z: https://upload.wikimedia.org/wikipedia/commons/thumb/5/5d/Baltik3-screenshot4.PNG/220px-Baltik3-screenshot4.PNG

Programuj – na této úrovni žák již opravdu programuje, jelikož se mu rozšíří možnosti ikonických příkazů a dalších parametrů nutných k programování. Žáci mají na výběr ze dvou režimů. V režimu *začátečník* je panel omezen pouze na základní příkazy, v *pokročilém* režimu se panel plně rozvine a zobrazí všechny možnosti, které program nabízí. Soubor příkazů Baltíka 3 zahrnuje všechny podmíněné příkazy (if, else if, switch-case), cykly (for, while, do-while), logické, relační a bitové operátory, proměnné, pole, procedury, rekurzi, příkazy pro práci se složkami a soubory, pro práci s datem a časem, pro práci s klávesnicí a myší, pokročilé grafické příkazy, matematické funkce atd. Pro ladění programů je k dispozici krokování, zobrazení proměnných. (SGP Baltík 3, 2017). Hlavně díky poslední funkci se z Baltíka stává plnohodnotný programovací jazyk, jelikož u většiny „dětských“ programovacích jazyků je fáze ladění a krokování velmi omezena nebo úplně opomenuta.

Obrázek 3 - Baltík - ovládací panel: Programuj pokročilý
zdroj: [online]. In: . [cit. 2019-07-07]. Dostupné z: http://www.zsalsova.cz/stare/i-centrum/baltik/baltik-2-01_soubory/image008.jpg

K programu je velmi dobře zpracována metodika výuky dostupná na stránkách tvůrců a na internetu je k dispozici nepřeberné množství návodů.[15]

Pokud je pro výuku programování využita nová verzi Baltie 4 C#, prostředí bude obohaceno o třetí rozměr, což bude pro žáky přeci jen atraktivnější. Výsledná grafika je bohužel i tak velmi primitivní, a ve srovnání s 3 D grafikou, kterou žáci znají z různých novodobých her, působí tato velmi zastarale. Přínosem je volba zobrazení C# kódu, kde mohou starší žáci porovnávat grafický zápis se skutečným objektově orientovaným zápisem v jazyku C#.

Přestože je tento programovací jazyk zatížen počítačovou licencí, nelze opomenout velmi vstřícný krok autorů směrem ke školství. V rámci výhodné multilicenční politiky lze využívat libovolnou verzi programovacího jazyku společnosti SGP Systems. Licence je bohužel časově omezena pouze na 1 rok.

4.1.6 Kodu Game Lab

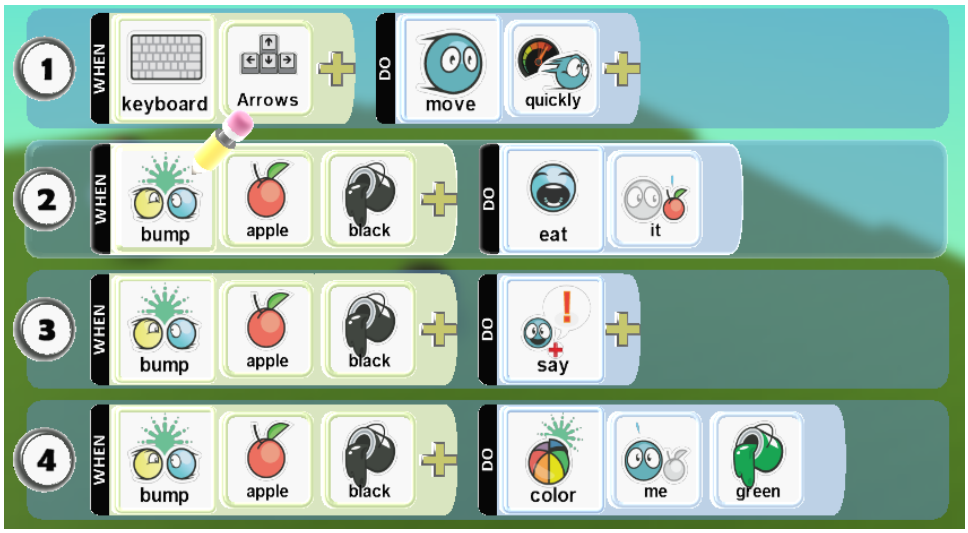
Společnost Microsoft před nedávnem vyvinula jednoduché prostředí, ve kterém si lze během chvíle s minimálními znalostmi programovacího kódu vytvořit vizuálně působivé hry. Tento program se lehce vymyká klasickému výukovému softwaru, jelikož jeho

[15]SGP Systems - Baltík C# 3D vizuální výukové programovací nástroje pro děti, mládež a dospělé [online]. [cit. 2019-07-05]. Dostupné z: https://www.sgpsys.com/cz

původní zaměření bylo cíleno pouze na tvorbu počítačových her. Z tohoto důvodu je ovládání programu navrženo pro herní ovladač stejnojmenné firmy. Přesto ho lze provozovat také v počítači.

Celý systém je postaven na výběru vhodného příkazu z nabídky dlaždic umístěných na kruhových výsečích, pomocí kterých žáci postupně skládají výsledný program. Na základě volby určitého nástroje se poté žákům otevírá nabídka s dostupnými možnostmi nastavení konkrétního nástroje, což velmi zjednodušuje tvorbu kódu. Program obsahuje celkem cca 120 dlaždic, které mohou žáci ke svému programování využít.

Kodu obsahuje tři zásadní změny, které ho odlišují od programovacího jazyka Baltík. První změnou je čistě objektový přístup, se kterým se žáci setkávají již při prvních zkušenostech s programem. Zvykají si na to, že každý objekt má své vlastnosti, kterými se liší od ostatních. Druhou změnou je styl zápisu programového kódu, který se velmi nápadně podobá klasickému programování. Jednotlivé řádky kódu jsou uvozeny podmínkovým příkazem, který rozhoduje o tom, zda se daná část kódu bude vykonávat, či nikoliv. Tímto způsobem začnou žáci samovolně přemýšlet o jednotlivých akcích, které daný objekt vyhodnocuje, setkávají se s řízeným průběhem programu. Nevýhodou je, že se žáci nenaučí, jak tento zápis v klasickém programovacím jazyce napsat.



Obrázek 4 - Kodu Game Lab - detail programovacího prostředí
zdroj: [online]. In: . [cit. 2019-07-07]. Dostupné z: http://2.bp.blogspot.com/-UOmVQHFNP5Y/VFeF3etHtLI/AAAAAAAAAB4/iuflAuRR5OU/s1600/My%2BKodu.png

Třetím a rozhodujícím faktem ve školním prostředí je jeho dostupnost, což je velkou předností tohoto systému. Není vázán žádnou placenou licenční smlouvou. Součástí tohoto systému v ČR je navíc základní metodická příručka, která je vyvíjena v rámci stále trvajícího projektu Akademie programování[16], pomocí které mohou učitelé připravit celou hodinu (respektive několik vyučovacích hodin) díky čtyřem základním, dopodrobna zpracovaným programovacím tématům, k nimž mají pedagogové přístup po zaregistrování své školy do tohoto projektu. Díky této registraci získají zcela zdarma úvodní zaškolení zkušenými lektory. Tato aktivita vznikla na základě mezinárodního projektu Hour of Code (Hodina kódu)[17] , který nabízí didakticky zpracované vyučovací hodiny programování v několika zvolených programovacích jazycích, ve 45 cizojazyčných variantách včetně českého jazyka. Této metodiky mohou využít žáci také ke svému samostudiu a rozvíjení svých programátorských a algoritmizačních schopností.

Kodu má také svá omezení, která vychází ze zaměření systému, který je více orientován na grafickou stránku výsledné hry, méně na jeho funkční možnosti. Autor s tímto programem pracuje v rámci mimoškolní aktivity pro 3. až 5. ročník na Základní škole Ladova 5 v Litoměřicích. V rámci této aktivity se při práci na projektech narazilo na funkční omezení, se kterými se musí v návrzích dalších programů počítat, a je nutné je různými netradičními postupy alespoň částečně nahradit. Hlavním omezením je tvorba proměnných, která neumožňuje vytvářené proměné pojmenovat jako v jiných programovacích jazycích. V programu si musíte vystačit s předem definovanými proměnnými, kterými jsou score a zdraví. Dále jsou k dispozici předem definované globální score, které lze také k tvorbě proměné využít. Výhodou je možnost velkého množství označení proměnných typu score, ze kterých si lze udělat pomocné proměnné a využít je k testování různých situací.

[16] Akademie programování [online]. [cit. 2019-07-05]. Dostupné z: http://www.akademieprogramovani.cz/

[17] Learn. Připojte se k největší vzdělávací akce v dějinách, Prosinec 7-13, 2015 [online]. [cit. 2019-07-05]. Dostupné z: https://hourofcode.com/cz/learn

Dle názoru autora je systém Kodu, vzhledem k intuitivnímu přístupu a atraktivnímu grafickému zpracování, ideální pro první krůčky na cestě k programování. Jeho možnosti dalšího využití, hlavně u starších dětí, jsou značně omezeny.

4.2 Vybraní zástupci robotických stavebnic a programovatelných robotů

Programovatelní roboti a robotické stavebnice se vyznačují svou nezávislostí na počítači a lze je využívat i mimo učebnu informatiky. Práce s nimi však není zcela komfortní a pro složitější aplikace je téměř nepoužitelná, proto je tato oblast vyčleněna z metod, které jsou na počítači zcela nezávislé.

Vzhledem k rozvoji moderních technologií se do výuky algoritmizace a rozvoje programování čím dál tím častěji zapojují různé programovatelné elektronické hračky a stavebnice. Tato forma výuky je pro většinu žáků atraktivní, přesto se s ní mohou setkat pouze v rámci zájmových činností Domu dětí a mládeže s elektrotechnickým zaměřením. Je to z toho důvodu, že pořízení podobných pomůcek v dostatečném počtu je pro většinu základních škol velmi nákladné. Není to však nemožné, a proto budou v kapitole představeni zástupci patřící do této kategorie výukových pomůcek.

4.2.1 Logo[18]

Přestože trend robotických stavebnic a robotů je na vzestupu nyní, lze podobný přístup vypozorovat již v 70. letech 20. století. Původcem těchto robotů byla mechanická želvička Logo vyvinutá Tomem Callahanem na MIT v letech 1969-1970. Tento robot byl ovládán stejnojmenným programovacím jazykem. Celá koncepce se postupem času zdokonalovala, z původního kabelového přenosu informací se přešlo na bezdrátový přenos, a došlo k designové změně podoby robota. Součástí robota byla zápisová část, která mohla zaznamenávat trasu robota, čímž se z něj stal grafický nástroj. Ve spojení s touto koncepcí vešel do povědomí pojem *želví grafika.* Vzhledem k omezeným možnostem pohybu vykazuje tato grafika typické znaky. Stejné koncepce se využilo

[18][1] 1969 – The Logo Turtle – Seymour Papert et al (Sth African/American). Cyberniczoo.com: a history of cybernetic animals and early robots [online]. [cit. 2017-07-07]. Dostupné z: http://cyberneticzoo.com/cyberneticanimals/1969-the-logo-turtle-seymour-papert-marvin-minsky-et-al-american/

i v případě převodu do virtuálního prostředí. Přestože je tato koncepce velmi stará, navzdory tomu je této technologie využíváno dodnes v každém dálkově ovládaném zařízení. Jediný rozdíl mezi původním a stávajícím pojetím je využití soudobé technologie, nabízející např. bezdrátový přenos.

Existuje celá řada podobných robotických „stavebnic“, většinu z nich lze využít také pro výuku programování u starších dětí, bohužel se ve všech případech jedná o placené řešení, na školní poměry většinou poměrně finančně nákladné.

4.2.2 Cubetto[19]

> *„Cubetto je přátelský robot, který naučí vaše děti základům programování a logického myšlení bez použití počítače.“ (Malý programátor, 2017)*

Zcela opačný přístup zvolili tvůrci systému Cubetto, který k tvorbě programu ovládajícího dřevěného robota využívá speciální programovací panel, který je bezdrátově propojen se samotným robotem. Hlavní výhodou tohoto systému je samotný zápis programu, u kterého není podmínkou zvládnutí četby. Jednotlivé příkazy se provádí pomocí vkládání různobarevných dřevěných dílků do programovacího panelu. Příslušný příkaz je intuitivně reprezentován příslušným tvarem. Skládáním jednotlivých dílků žáci vytváří frontu příkazů, což je základním konceptem programování. Žáci mohou použít pouze 4 příkazy, z čehož jeden slouží pro pohyb kupředu, další dva ke změně směru a poslední využívá volání funkce, která se nachází ve spodní části panelu a obsahuje celkem čtyři pozice.

[19] Malý programátor [online]. [cit. 2017-07-07]. Dostupné z: https://www.malyprogramator.cz/

Obrázek 5 - Cubetto- detail programovacích dílků
zdroj: [online]. In: . [cit. 2019-07-07]. Dostupné z: https://media.canadianclassroom.com/catalog/product/cache/1/thumbnail/c96a280f94e22e3ee3823dd0a1a87606/c/u/cubetto-blocks.jpg

Robot ve tvaru dřevěné kostky se pohybuje po hracím plánu a cílem je dostat se z jednoho místa hracího pole na druhé. Celý systém je v dnešní době dostupný v několika českých obchodech.

4.2.3 Bee-bot[20]

Edukační interaktivní pomůcka Bee-bot neboli robotická včelka je nápadně podobná mechanické želvě Logo. Jedná se o mechanického robota, který se ovládá přímo a je programovatelný. K programování se využívá tlačítek umístěných na zádech včelky, pomocí nichž lze předem určit její pohyb.

[20] Bee Bot Včelka - interaktivní robot [online]. [cit. 2019-07-07]. Dostupné z: https://www.vyuka-vzdelanie.sk/bee-bot-vcelka.html

Obrázek 6 - Detail ovládacích prvků
zdroj: [online]. In: . [cit. 2019-07-07]. Dostupné z:
https://www.sammat.education/files/uploads/2015/07/ITSBOT_web3.png

Základem této didaktické pomůcky je čtvercová síť. Tzv. krok včelky tvoří čtverec o rozměrech 15 x 15 cm. Vnitřní paměť včelky dokáže pojmout jen 40 kroků, což souvisí také s velikostí podložky. Bee-bot se dá využít i v předmětech nezabývajících se informatikou, jen stačí jeho podložku tématicky přizpůsobit danému předmětu. Vzhledem k vizuálnímu vzhledu a možnosti pouhého lineárního programování, je tento robot vhodný pro předškolní vzdělávání či domácí rozvoj algoritmizace v nižším věku.

4.2.4 Lego WeDo v2.0[21]

Nejnovějším přírůstkem do rodiny robotických hraček je Lego WeDo v2.0, kterému je věnována tato kapitola.

Hlavním důvodem vzniku systému Lego WeDo v2.0 je jeho podobnost se systémem Lego Mindstorms. Pro počáteční seznámení s robotickou hračkou není nutné hned pořizovat komplexní systém Lego Mindstorms, ale naprosto postačí právě systém Lego WeDo v2.0.

Hlavní rozdílem jsou menší možnosti v sestavení robota, jelikož řídící jednotka[22] je osazena pouze dvěma vstupy. Také samotné senzory nemají tak bohatou nabídku jako v případě Lego Mindstorms. WeDo ve své sadě nabízí pouze dva senzory, a to pohybový senzor a senzor náklonu.

[21] LEGO Education WeDo 2.0 Core Set [online]. [cit. 2019-07-07]. Dostupné z: https://education.lego.com/en-us/products/lego-education-wedo-2-0-core-set/45300
[22] Řídící jednotka systému Lego WeDo je označována jako Smart Hub. Doplňkovou komponentou je dobíjecí baterie, která se s řídící jednotkou spojí. Díky ní je zajištěno napájení na několik hodin.

Obrázek 7 - SmartHub s napájecí jednotkou
zdroj: [online]. In: . [cit. 2019-07-07]. Dostupné z: https://le-www-live-s.legocdn.com/images/423923/live/sc/Products/45301/45301_Prod_02/950656655854566e0491a5df982c5db1/05ce4f6d-7040-4aa8-8e5c-a56200c62d71/original/05ce4f6d-7040-4aa8-8e5c-a56200c62d71.jpg?fit=inside|855:640

Systém Lego WeDo je primárně zaměřen na mladší školní věk a jeho hlavním cílem je seznámit děti se světem robotiky a základy programování robotů. Myšlenka je podpořena designovými doplňky stavebnice, které dělají z robotů více „dětsky“ vypadající roboty. Dalo by se říci, že se jedná o zjednodušenou verzi předchozího systému. Přes všechna tato omezení lze postavit mnoho rozličných robotů.

Obrázek 8 - Vývojové prostředí WeDo 2.0
zdroj: https://store-images.s-microsoft.com/image/apps.60030.13510798886702495.78c928f4-152e-428b-a979-60b18b38d63a.2e6ace16-d148-4a77-ba98-e10fe5f31990?w=672&h=378&q=80&mode=letterbox&background=%23FFE4E4E4&format=jpg

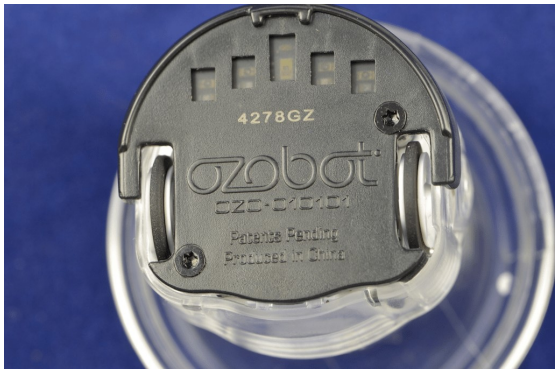
K programování řídící jednotky lze použít programové prostředí, které je volně dostupné na stránkách výrobce. Prostředí je jednoduché a pro děti intuitivní díky názornému zobrazení jednotlivých ovládacích prvků. Chybí česká lokalizace vývojového prostředí, což je kompenzováno názorným grafickým zpracováním, což usnadňuje sestavování programu. Užitečná je integrace knihovny možných projektů, každý s podrobným návodem na stavbu a nastavení robota, včetně metodicky zpracovaných pracovních listů. Pokud žáci ovládají programové prostředí Scratch, lze do něj doinstalovat doplněk, který přidá speciální sadu příkazů k ovládání komponent WeDo.

Propojení vývojového prostředí a řídící jednotky robota je zajištěno pomocí bluetooth technologie, která umožňuje ovlivňovat chování robota v reálném čase a odpadá složité připojení k USB kabelu. Nabízí se proto varianta použití jedné stavebnice ve třídě. Jednotlivé vývojové týmy vytvoří svou aplikaci na PC a pro otestování ji stačí jen nahrát do řídící jednotky. Vytváří se tak kreativní prostředí pro práci vývojových týmů, podmínkou je vývoj aplikace pro předem postaveného robota, s jasně daným omezením funkčnosti a možnostmi využití. Vzhledem k počtu participujících skupin ve třídě není možné vyzkoušet funkčnost aplikace v průběhu vývoje.

4.2.5 Ozobot[23]

Tento malý robot, kterého lze dnes bez problémů pořídit v několika českých obchodech, je ve skutečnosti „sledovač“ čáry. Sledování čáry je ve své podstatě jednoduchá úloha. Obtížnou se stává v případě, že je nutný plynulý a rychlý pohyb robota po vytyčené trase. Robot je řízen pomocí tzv. *ozokódu*, který je složen ze čtyř barev, konkrétně černé, modré, zelené a červené, kdy každá barva konkrétním způsobem mění chování robota. Barvy jsou z herního plánu načítány díky soustavě pěti světelných čidel, které jsou umístěny vpředu ve spodní části podvozku.

[23] Ozobot ve výuce - robůtci s českým <3 [online]. [cit. 2019-07-05]. Dostupné z: http://ozobot.sandofky.cz/



Obrázek 9 - Detail podvozku ozobota s fotosenzory
zdroj: [online]. In: . [cit. 2019-07-07]. Dostupné z: https://i1.wp.com/robotfun.co.uk/wp-content/uploads/2014/12/ozobot-bottom.png?fit=1024%2C678&ssl=1

Příkaz pro robota, tzv. *ozokód*, se skládá ze sestavy několika konkrétních barev v úsecích dlouhých 5-7 mm pro každou z nich. Takto se dá jednoduše ovlivňovat pohyb ozobota po vyznačené trase. Trasa nemusí být nakreslená pouze na papíře, v době moderních technologií lze k zakreslení využít různá zobrazovací dotyková zařízení, například tablet.

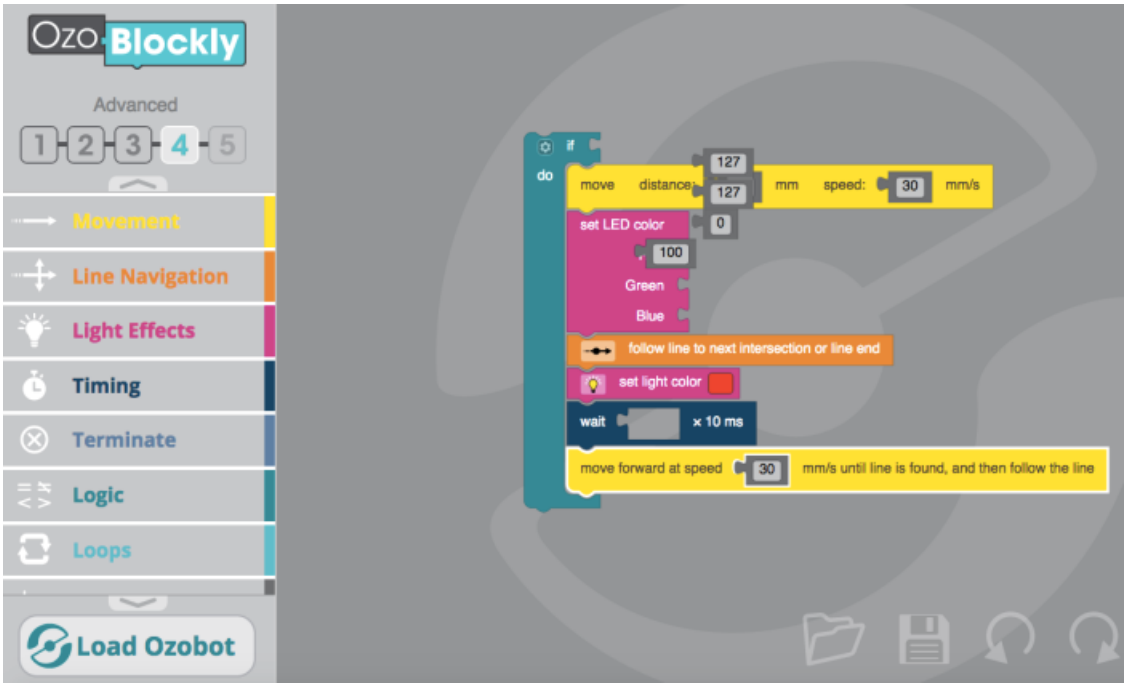
Obrázek 10 - Ukázka ozokódu tvořený barevnou sekvencí
zdroj: [online]. In: . [cit. 2019-07-07]. Dostupné z: http://www.dtpko.cz/wp-content/uploads/2015/12/Ozobot-OzoCodes-Reference-1546x1200.jpg

Pokud se má robot pohybovat po trase vyznačené na papíře, je třeba zvolit barvy fixů ve správných odstínech, aby je robotova čidla dokázala správně interpretovat. Nejvhodnějším typem popisovačů jsou fixy od společnosti Centropen, pro záznam černé a červené barvy se využívají permanentní popisovače této značky, a pro záznam zelené a modré jsou vhodnější popisovače na bílou tabuli. K novější verzi ozobota evo jsou dnes dodávány speciální popisovací fixy. Kromě správného odstínu barvy je důležité dodržet také minimální šířku čáry 5-6 mm, proto by používané popisovače měli mít skosený hrot. Pokud je k dispozici pouze popisovač se špičatým hrotem, dá se nakreslit dvojitá čára.

Pokud bude k pohybu ozobota využita obrazovka tabletu, lze pro kreslení čar využít aplikačního prostředí OzoDraw, které je podporováno všemi platformami mobilních OS. Nabízí také kompletní přehled všech příkazů, které lze libovolně umisťovat na vytvořenou trasu.

Kromě využití tzv. *ozokódu*, který vzniká příslušnou sekvencí barevných značek, lze robota programovat pomocí multiplatformního aplikačního prostředí *OzoBlockly*, ve kterém jsou využity předdefinované nabídky příkazů rozdělených do logických celků, díky nimž je možné programovat pohyb robota bez přítomnosti vodících čar na ploše. Prostředí je nápadně podobné programovacímu jazyku Scratch. Programové prostředí, rozdělené na logické celky (viz. obr. 11), ovládá nejenom pohyb robota, ale dokáže také ovlivňovat světelné efekty tvořené soustavou RGB LED diod, které jsou umístěné v horní části robota pod průhledným krytem.

Převedení hotového programu do robota probíhá bezdrátově, přiložením spodní části robota na vyznačené místo na obrazovce. Takto naprogramovaný robot může plnit mnohem sofistikovanější úkony, kterých nelze dosáhnout pouhým zápisem grafických značek.



Obrázek 11 - Programové prostředí OzoBlockly k programování robota
zdroj: [online]. In: . [cit. 2019-07-07]. Dostupné z:
https://stevebrophy.files.wordpress.com/2015/08/screen-shot-2015-08-15-at-4-12-00-pm.png

Ozobota lze pořídit v několika verzích lišících se od sebe převážně senzorovým vybavením. Základní verze Ozobot v2.0 obsahuje pětici světlocitlivých čidel umístěných na podvozku robota, nová verze Ozobot Evo je rozšířena o sedm čidel a senzorů, které nově detekují situaci nejen pod ním, ale také kolem robota. Toto chování je zajištěno díky vnitřní inteligenci robota. Evo dokáže například rozpoznat překážku před ním a zastavit, aniž by do ní naboural.

Jelikož tvůrci ozobota cílí na vzdělávací instituce, ne na domácnosti, vytvořili pro pedagogy rozsáhlý portál [24] shromažďující rozličné aktivity a nápady pro výuku. K inspiraci a jako zdroj dalších informací o této didaktické pomůcce lze využít také české stránky, zabývající se využitím Ozobota v edukačním procesu.[25]

Předností této koncepce je možnost libovolného vzhledu robota, což lze realizovat v hodinách pracovních činností, a vytvořit tak zajímavou mezipředmětovou vazbu. Na robota lze „navléknout“ libovolný vzhled vytvořený z papíru, který nemá vliv na

[24]Ozobot user portal [online]. [cit. 2019-07-05]. Dostupné z: http://portal.ozobot.com/lessons
[25] Ozobot ve výuce - robůtci s českým <3 [online]. [cit. 2019-07-05]. Dostupné z: http://ozobot.sandofky.cz/

funkčnost, neboť jediná funkční část ve spodní části robota není převlekem dotčena. Konstrukce je natolik lehká, že jeho hmotnost neovlivňuje pohyb robota. Na internetových stránkách výrobce existuje bezpočet předloh, pomocí nichž lze změnit vzhled ozobota.[26].

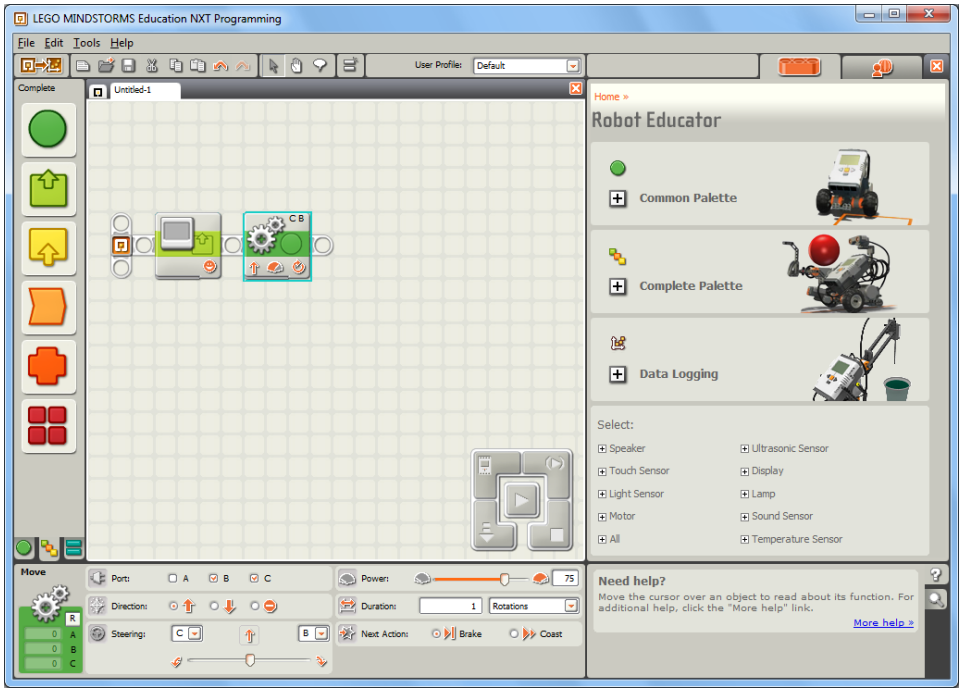
4.2.6 Lego Mindstorms[27]

Lego Mindstorms, předchůdce zjednodušené verze Lego WeDo 2.0, je určen pro pokročilejší stavbu programovatelných robotů. Oproti zjednodušené verze nabízí větší množství příslušenství a stavebních prvků, díky kterým lze tvořit i rozsáhlejší projekty.

Přestože lze k programování využít vestavěné sady příkazů v centrální jednotce stavebnice, není tato manipulace příliš komfortní, a mnohem snadněji lze stavebnici programovat skrze počítačový program nabízející mnohem širší škálu dostupných příkazů.

Základem celé stavebnice je řídící jednotka tzv. *kostka*, která vyhodnocuje informace z připojených senzorů, díky nimž řídí aktivní prvky stavebnice. Pro jednodušší programování lze využít volně dostupné programové prostředí, které je postaveno na stavebnicové koncepci a je silně intuitivní.

[26]Connected and Screen-Free Games | Ozobot [online]. [cit. 2019-07-05]. Dostupné z: http://ozobot.com/play/print-games
[27] O EV3 - Mindstorms LEGO.com [online]. [cit. 2019-07-07]. Dostupné z: https://www.lego.com/cs-cz/mindstorms/about-ev3



Obrázek 12 - Programovací prostředí
zdroj: [online]. In: . [cit. 2019-07-07]. Dostupné z: https://lego.zcu.cz/web/images/stories/nxt_software/app/printscreen_nxt_programming.png

Každý blok příkazů má dodatečná nastavení, která jsou umístěna v dolní části prostředí a slouží k přesnějšímu nastavení jednotlivých senzorů. Bohužel citlivost některých senzorů je problémem, na který se musí při stavbě robota pamatovat. Jedná se o druhy fotosenzorů, které jsou ovlivňovány osvětlením okolního prostředí. Předností tohoto systému je možnost volby programovacího jazyka. Pro různé programovací jazyky lze nahrát do kostky pomocí speciálního firmwaru příslušný překladač. Další výhodou je možnost univerzálního propojení různých Lego Mindstorms a jeho prvků, stejně jako je TOMU zvykem u klasické Lego stavebnice. Díky této univerzálnosti lze stavět nejen funkčně působivé, ale také vizuálně atraktivní samočinné jednotky.

Hlavní překážkou rozšíření na běžném typu základních škol je bezesporu její pořizovací cena, která přesahuje 10 000 Kč. Stavebnice je hojně využívána v základním školství se zaměřením na IT technologie, při zájmových činnostech a mimoškolních aktivitách.

4.3 Vybraní zástupci herních prostředí

Hry mají pro děti a žáky největší přínos, protože k edukaci dochází mimovolně, v jejich přirozeném a dobře známém prostředí, a proto k nim přistupují s nadšením. To samé platí pro hry a herní prostředí v rámci výuky algoritmizace, kdy žáci nemusí mnohdy ani postřehnout, že k edukaci dochází, vymyslet řešení problému je bráno jako součást celé hry. I zde jsou jistá omezení, díky kterým nejsou herní prostředí při výuce tak často využívána.

Prvním omezením jsou finanční možnosti školy. Vetšina herních prostředí je totiž licenčně vázána. Druhé omezení se týká nízkého počtu hodinových dotací ve výuce, kdy je třeba žákům předat v relativně krátkém čase větší množství informací, a nelze vždy „čekat“, až jednotlivé vztahy a zákonitosti v rámci hry objeví sami. I přesto jsou herní prostředí čas od času vhodná jako zpestření vyučovacích hodin, které žáci vřele uvítají.

Nejrozšířenější a nejznámější herní prostředí na našich školách je Minecraft, a druhým je programovací portál CodeCombat, který lze, s jistým omezením, využít ve školním prostředí bezplatně.

4.3.1 Minecraft[28]

Minecraft se do škol dostal v roce 2016 díky speciální edici Minecraft: Eduction Edition. Tuto edici je možné v rámci školního prostředí zdarma vyzkoušet, nicméně integrace Minecraftu do výuky je již zpoplatněna. Do vývoje této edice se v rámci testování zapojilo přes 50 000 studentů a učitelů.

Základem celé této hry je tvorba rozličných objektů z předem natěžených surovin. Tato strategie je však nemyslitelná ve školním prostředí díky zdlouhavé fázi edukace, a proto se v edici Education od tohoto principu ustoupilo, a žáci mají k dispozici neomezené zdroje. Mohou tedy bez jakýchkoliv zdržení začít řešit zadané úlohy, které přímo v tomto prostředí vytváří učitel. Úkolem žáků je na základě posloupnosti instrukcí zautomatizovat práci, kterou v běžné herní edici řeší většinou manuálně.

[28]MICROSOFT TECHNET. Minecraft: Education Edition je tady, vyzkoušejte si školu hrou [online]. In: . 2. prosince 2016 [cit. 2019-07-09]. Dostupné z: https://www.zive.cz/bleskovky/minecraft-education-edition-je-tady-vyzkousejte-si-skolu-hrou/sc-4-a-185111/default.aspx

4.3.2 CodeCombat

CodeCombat je unikátní portál, který je vhodný pro výuku algoritmizace jak v rámci zájmových činností, tak ve vyučovacích hodinách na ZŠ, což bylo autorem práce v rámci jeho pedagogické praxe odzkoušeno. Největší výhodou je herní systém, ve kterém je hrdina ovládán na základě psaného programovacího kódu. Tento systém je pro žáky vhodný, neboť programování funguje na bázi hry. Žáci vylepšují svého hrdinu na základě systému odměn, a díky tomu snadnější postupují do vyšších úrovní.

K zapojení do systému je nutná registrace pedagoga i všech žáků. Pedagog vytvoří tzv. třídu, do které se žáci zaregistrují bez použití emailové adresy, pouze s pomocí speciálního kódu. Pedagog má poté přehled o tom, jak žáci postupují jednotlivými urovněmi a má možnost kontrolovat zápisy programového kódu jednotlivých žáků. Při zakládání tzv. třídy má pedagog možnost zvolit jeden ze dvou nabízených programovacích jazyků, kterými jsou jazyk Python a JavaScript. Volba konkrétního jazyka je na uvážení pedagoga, přesto autoři[29] doporučují pro výuku úplných záčátečníků programovací jazyk Python. Důvodem je jeho jednodušší syntaxe. Pro pokročilejší žáky je zde připraven programovací jazyk JavaScript, hojně využívaný při tvorbě webových aplikací.

Herní prostředí je dobře přizpůsobeno žákům, kteří si mohou sami zvolit své individuálního tempo práce a přizpůsobit ho svým znalostem a zkušenostem. Jednotlivé úrovně lze plnit bez pomoci pedagoga díky propracovanému systému nápověd. Na začátku každé úrovně je pro žáky připraven podrobně rozebraný cíl se stručným návodem. Pokud se žáci při plnění úlohy dostanou do problémů, mohou využít tlačítka nápovědy, které zobrazí podrobnější rozbor dané problematiky a nasměruje žákovo myšlení správným směrem. Jak již bylo zmíněno, odměnou za správné řešení jsou body a drahokamy, které mohou žáci směnit za různá vylepšení pro své hrdiny. Pokud žáci splní celý soubor úkolů, získají osobní certifikát, což může být motivace k plnění dalších úloh.

Ač se může zdát, že je herní prostředí Code Combat vhodné pouze pro pedagogy, kteří mají s programováním zkušenosti a ovládají ho, autoři programu mysleli i na pedagogy,

[29] CodeCombat - Learn how to code by playing a game [online]. [cit. 2019-07-05]. Dostupné z: https://codecombat.com/teachers/resources/faq#what-is-codecombat

kteří tak erudovaní nejsou. Po registraci do systému dostane pedagog k dispozici řešení všech dostupných úrovní, včetně podrobného návodu a zápisu programového kódu, který by měli žáci vytvořit. Začlenění tohoto herního prostředí do výuky se tak nemusí bát ani pedagogové, kteří programování neovládají na výbornou.

Dle tvůrců[30] je toto prostředí vhodné od devátého roku, což představuje 4. ročník na ZŠ. Tato cílová skupina je především ovlivněna nutností vlastního zápisu programového kódu, ve kterém jim napomáhá nápovědní systém, který umožňuje pouze zápis části příkazu a následného výběru z dostupných příkazů, obsahujících tuto část kódu. V rámci postupného učení je soubor příkazů postupně rozšiřován, což umožňuje lepší orientaci v řešení dané problematiky. Před každou novou dovedností je zařazen systém tutoriálů, které žákům napomáhají k rychlejšímu pochopení dané dovednosti. Toto všechno napomáhá k větší samostatnosti a možnosti individuálního tempa.

Další skutečností, která napomáhá snadnějšímu začlenění do výuky, je online prostředí, které umožňuje pokračování v dalším průběhu výuky na jakémkoliv školním počítači, či počítači domácím v rámci samostatné přípravy. Jediný nedostatek v tomto systému autor shledává v absenci mobilní platformy, jako je android nebo iOS. Vzhledem k nutnosti zápisu programového kódu autorem je však tento nedostatek opodstatněn.

Celé výukové prostředí je rozděleno na několik celků, z čehož první celek, nazvaný *Úvod do výpočetních věd*, je možno vyzkoušet zcela zdarma. Pro využití dalších částí je nutné zakoupit licenci, která je individuálně spočítána na základě zaslání požadavků autorům programovacího prostředí.

Nespornou výhodou je objektivě orientované pojetí, které je v dnešních programovacích technikách preferované. I vzhledem k této skutečnosti je dané prostředí vhodné pro rozvoj algoritmizace a programování.

V rámci ověření tohoto prostředí ve školní praxi, autor zařadil tuto výuku do informatiky v 6. ročníku a v rámci zájmové činnosti na I. stupni ZŠ. V obou případech se ukázalo, že žáci systém výuky velmi rychle pochopili, k čemu do značné míry napomohlo převážně lokalizované prostředí, a byli schopni uspěšně plnit jednotlivé úlohy. Jelikož se příkazy

[30] CodeCombat - Learn how to code by playing a game [online]. [cit. 2019-07-05]. Dostupné z: https://codecombat.com/teachers/resources/faq#what-is-codecombat

zapisují v angličtině, u malé části žáků tato skutečnost z počátku tvořila jisté obtíže, které však byly v průběhu hry odbourány. Jedinou nevýhodou úvodního kurzu je jeho nízká časová dotace, takže zkušenější žáci jsou schopni tuto oblast splnit v rámci jedné vyučovací hodiny. Jelikož autor neměl možnost vyzkoušení placené verze, další popis bude čerpán pouze z webových stránek projektu.

V rámci licencované verze se žáci mohou naučit tvořit počítačové hry, vytvářet webové stránky či pokračovat v dalších úrovních výpočetních věd. Každá oblast je rozdělena do šesti úrovní. Obtížnost se stupňuje a rozšiřují se získané dovednosti nabyté v předchozích úrovních.

Pro pedagogy jsou na webu k dispozici studijní materiály a metodiky, které pomáhají zařazení tohoto prostředí do výuky. Velkým přínosem je také orientace na projektové vyučování, které je u žáků velmi oblíbené.

4.4 Pedagogická opora výuky programování

V této kapitole je představeno několik metodických portálů pro pedagogy, které mohou být nápomocné při zavádění algoritmizace a programování do výuky informaticky zaměřeného předmětu na ZŠ. Vzhledem k rozsáhlým možnostem těchto portálů je uveden pouze přehled se stručným popisem. Velkým přínosem zmiňovaných portálů je skutečnost, že výuka algoritmizace a programování není zaměřena pouze na jedno programovací prostředí či jazyk. Portálů s obdobným zaměřením či online prostředí existuje mnohem více, vzhledem k zaměření práce jsou zvoleny pouze autorem ověřené zdroje.

- **Kidscodr**[31] – Portál nabízí rozličné výukové kurzy formou videotutoriálů, které jsou pro žáky nejnázornější. Do těchto kurzů se může žák zaregistrovat zdarma, např. pomocí google účtu nebo účtu na Facebooku. Veškeré kurzy jsou rozděleny do dvou částí. V první části je žákovi ukázána příslušná programová dovednost, kterou později využije při plnění úkolu, který je žákovi zadán vždy na konci každého bloku kurzu. V rámci registrace do kurzu získá žák veškeré potřebné materiály, které pro splnění úkolu potřebuje. Díky tomu se žáci mohou věnovat

[31] Učíme děti programovat | KidsCodr [online]. [cit. 2019-07-05]. Dostupné z: https://www.kidscodr.cz/

pouze programové části zadaného úkolu, což velmi pozitivně ovlivňuje časovou náročnost splnění úkolu.

- **Hour of Code**[32] – Portál poskytuje pedagogům soubor metodicky zpracovaných hodin programování v rozličných programovacích prostředích. Poskytuje návody pro zapojení programování do výuky, nejen pro pedagogy informatiky, ale také pro zapojení do mimoškolního vzdělávání

- **Informatické myšlení**[33] – Portál zastřešuje pilotní projekt PRIM. Sdružuje výukové materiály pro pedagogy zavádějící algoritmizaci a programování do výuky na ZŠ

- **Akademie programování**[34] – Portál sdružuje výukové aktivity pro žáky i metodicky zpracované materiály pro pedagogy. V rámci žákovského samostudia je vytvořen soubor videotutoriálů, které napomáhají ke snadnějšímu zvládnutí danných úkolů.

[32] Připojte se k největší vzdělávací akce v dějinách, Prosinec 7-13, 2015 [online]. [cit. 2019-07-05]. Dostupné z: https://hourofcode.com/cz/
[33] Informatické myšlení: [online]. 2017 [cit. 2019-07-04]. Dostupné z: https://www.imysleni.cz
[34] Akademie programování [online]. [cit. 2019-07-05]. Dostupné z: http://www.akademieprogramovani.cz

4.5 Konkrétní přílady informaticky zaměřených aktivit

Tato kapitola obsahuje zpracované metodiky ke dvěma aktivitám, které vycházejí z ověřených postupů aplikovaných v rámci vedení zájmového kroužku se zaměřením na programování.

Grafické programování na papíře

Třída: 5. ročník

Časová dotace aktivity: 45 minut

Cíle výuky:

- Žák sestaví kód pomocí předem dohodnutých symbolů
- Žák vytvoří vlastní kód pro spolužáka
- Žák chápe obtížnosti převodu reálných problémů do programu
- Žák si uvědomí rozdílný přístup v myšlením člověka a v interpretaci stejného algoritmického postupu počítačem

Prekoncept:

- žák využije informací z výkladu pedagoga a vlastní znalost problému např. z počítačových her

Pomůcky:

- pracovní list s připravenou mřížkou 4x4
- předloha s vytvořenými úkoly

Postup:

- Na začátku této aktivity učitel vysvětlí žákům pojmy algoritmus a program.
- Učitel žákům rozdá pracovní listy s „hracími poli“ a předem dohodnuté používané symboly programového kódu (viz příloha č. 1a a č. 1c).
    - Při volbě rozsahu symbolů je třeba mít na paměti, že větší počet symbolů dává žákům větší variabilitu při řešení daného problému, díky níž je hledání správného řešení jednodušší.
    - Naopak zúžením tohoto výběru jsou žáci nuceni ke stavbě optimalizovaného programového kódu.

- Žák převede příslušný obrazec do symbolického zápisu.
    - Pokud žák objeví v symbolickém zápisu opakující se části, přepíše tento zápis v kombinaci číslice a příslušného symbolu (pro žáka je důležité si tento postup osvojit a zapamatovat, bude využit jako prekoncept v úvodu výuky o cyklech).
- Žák předá tento zápis jinému žákovi, který na základě sepsaného postupu, který dodržuje, sestavuje stejný obrazec.

Poznámka:

- Výhodou této aktivity je možnost jejího využití i mimo hodiny informatiky, protože nevyžaduje použití počítače.
- Ná základě zadané předlohy lze vytvořit obdobné nebo i větší předlohy.
- Obdobnou metodu můžeme použít i v případě jiných povolených příkazů (např. je možné povolit pouze dva příkazy – o krok dopředu a otočení doprava).

Reflexe:

Tato aktivita byla ověřena v rámci zájmové činnosti žáků 5. ročníku ZŠ. Aktivita byla zařazena při úvodní lekci programování, se kterým většina žáků neměla žádné předchozí zkušenosti. Náplň aktivity byla žáky velmi rychle pochopena, a v případě široké škály příkazů žáci bez problému plnili zadané úlohy. Po zredukování počtu příkazů a požadavku zkráceného zápisu (opakující se sekvence byly doplněny o hodnotu, která definovala počet opakování) nastaly žákům obtíže při zápisu programového kódu, jelikož si z počátku neuvědomovali, jak velké části programového kódu se opakují. Když si žáci celý programový kód napsali v nezkrácené verzi, byli poté schopni identifikovat stejné části kódu a zredukovat ho do zkrácené podoby. Také tvorba zápisu kódu, bez konkrétní předlohy, která představovala ovládaný objekt, byla z počátku mírně obtížná.

Tuto aktivitu lze různě obměňovat, vzhledem k prostředí, ve kterém je aktivita uskutečněna a také vzhledem k osobní invenci pedagoga.

Lidský automat

Třída: 5. ročník

Časová dotace aktivity: 45 minut

Cíle výuky:

- Žák se naučí používat předem připravené příkazy
- Žák využívá informačního myšlení při tvorbě sekvencí příkazů

Prekoncept:

- Žák využije své znalosti z počítačových her

Pomůcky:

- min. 6 plastových kelímků
- dřevěné hůlky nebo špejle

Postup:

- Kelímky jsou vyskládány do řady na lavici nebo stůl.
- Žáci vytvoří dvojice a rozdělí si role – jeden se stává operátorem a druhý robotem.
- Operátor s použitím předem dohodnutých příkazů ovládá „robota“. Operátor tvoří sekvenci příkazů, na základě které robot mění svou pozici. Cílem je, aby robot postavil pyramidu z kelímků.
- Žák v roli robota, na základě operátorových příkazů, přemisťuje pomocí hůlek jednotlivé kelímky.

Reflexe:

Tato aktivita byla ověřena v rámci výuky v povinně volitelném předmětu Informatika na ZŠ. Byla zařazena jako motivační úloha, která vedla k uvědomění si pořadí úkonů, který robot při své činnosti koná. Na základě této vědomosti bylo úkolem žáků sestavit „ovládací program“ pro lidského robota, který měl za úkol postavit pyramidu

z plastových kelímků. Aktivita byla žáky velmi dobře přijata a vzhledem k sekvečnímu způsobu ovládání, nepůsobila většině žákům obtíže.

Aktivitu lze obměnit na základě aktuálních pomůcek, či konkrétnímu zadání úlohy.

5 Závěr

V bakalářské práci se zabývám výukou algoritmizace a programování na základních školách, neboť by se v dohledné době měla stát nedílnou součástí výuky informatiky v rámci vládní směrnice Strategie digitálního vzdělávání do roku 2020.

Vycházel jsem z výzkumů, které se uskutečnily v letech 2006-2007 a v roce 2013 na Katedře informačních technologií a technické výchovy na Pedagogické fakultě Univerzity Karlovy a mapovaly tehdejší stav výuky algoritmizace a programování na ZŠ. Jejich výsledky odhalily, že na základě nedostatečné aprobovanosti pedagogů je výuka algoritmizace a programování odsouvána do pozadí, případně není vyučována vůbec.

Bakalářská práce analyzuje různé metody a přístupy na podporu informatického myšlení u žáků, aby se zamezilo tomu, že je pedagogové hned na začátku odradí nevhodně zvolenými metodami nebo neatraktivním způsobem výuky.

V první části práce jsou vysvětleny základní pojmy, spojované s problematikou algoritmizace a programování. Druhá část práce se zabývá popisem jednotlivých metod a přístupů ve výuce tak, aby reflektovaly potřebu rozvoje nové kompetence u žáků v rámci počítačové gramotnosti. Ke všem metodám jsou vybráni vhodní zástupci. Při výběru některých zástupců jsem vycházel z vlastních zkušeností s těmito prostředími, které jsem získal v rámci dlouhodobého vedení zájmové činnosti se zaměřením na algoritmizaci a programování. K tzv. unplugged metodám jsou zpracovány metodické pokyny a pracovní listy, které byly v rámci mého vedení zájmové činnosti ověřeny ve výuce.

Všechny cíle bakalářské práce byly naplněny.

6 Seznam použité literatury

[1] BROMOVÁ, Jana. Výuka algoritmizace na základní škole - aktuální stav. Č. Bud., 2012. diplomová práce (Mgr.). JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH. Pedagogická fakulta

[2] ČERNOCHOVÁ, Miroslava. *Rozvoj informačně technologických kompetencí na základních školách: výzkum stavu a struktury informačně technologické gramotnosti*. v Praze: České vysoké učení technické, 2013. ISBN 978-80-01-05407-9.

[3] DOSTÁL, Jiří. *Badatelsky orientovaná výuka: kompetence učitelů k její realizaci v technických a přírodovědných předmětech na základních školách*. Olomouc: Univerzita Palackého v Olomouci, 2015. ISBN 978-80-244-4515-1.

[4] Hlavní zjištění výzkumu. [ZPRACOVAL KOLEKTIV ŘEŠITELŮ POD VEDENÍM VLADIMÍRA RAMBOUSKA]. Výzkum informační výchovy na základních školách. Plzeň: Koniáš, 2007, s. 14. ISBN 8086948102.

[5] KOUBKOVÁ, A; PAVELKA, J. *Úvod do teoretické informatiky*. 3. vyd. Praha: MATFYZPRESS, 2003. ISBN 80-86732-03-7

[6] KVASIL, Bohumil, et al. Algoritmus. In Malá československá encyklopedie. Praha: Academia, 1984. s. 108.

[7] MOTYČKA, Arnošt. Algoritmizace. Brno: Konvoj, 1999. ISBN 80-85615-80-0.

[8] MÜLLER, Karel. *Programovací jazyky.* První vydání. Praha: ČVUT, 2002. 219 s.

[9] PRŮCHA, Jan. WALTEROVÁ, Eliška a Jiří MAREŠ. *Pedagogický slovník*. 7., aktualiz. a rozš. vyd. Praha: Portál, 2013, s. 185. ISBN 978-80-262-0403-9.

[10] PŠENČÍKOVÁ, Jana. Algoritmizace. Kralice na Hané: Computer Media, c2007.

[11] SAMKOVÁ, Libuše, 2011. Badatelsky orientované vyučování matematiky. In: Sborník 5. konference Užití počítačů ve výuce matematiky. České Budějovice: JČU, s. 336–341. ISBN 978-80-7394-324-0.

[12] STEPHENSON, Chris a Valerie BARR. Defining Computational Thinking for K-12. CSTA Voice. 2011, vol. 7, no. 2, pp. 3–4.

[13] *STRATEGIE DIGITÁLNÍHO VZDĚLÁVÁNÍ DO ROKU 2020*. In: . Dostupné z: http://www.vzdelavani2020.cz/images_obsah/dokumenty/strategie/digistrategie.pdf

[14] Tematické celky informatických výukových aktivit. [ZPRACOVAL KOLEKTIV ŘEŠITELŮ POD VEDENÍM VLADIMÍRA RAMBOUSKA]. Výzkum informační výchovy na základních školách. Plzeň: Koniáš, 2007, s. 215-238. ISBN 8086948102.

[15] VITOVSKÝ, Antonín. *Moderní slovník softwaru*. První vydání. Praha: AV software, 2006. 588 s. ISBN 80-901428-8-5.

[16] WIRTH, Niklaus. *Algoritmy a štruktúry údajov*. 2.vyd. Bratislava: Alfa, 1989. ISBN 80-05-00153-3.

[17] ŽÁK, Petr. Kreativita a její rozvoj. Brno: Computer press, 2004. 315 s. ISBN 80-251-0457-5. Kapitola Systémy kreativního řešení problému, s. 124-126

7 Seznam internetových zdrojů

[1] 1969 – The Logo Turtle – Seymour Papert et al (Sth African/American). Cyberniczoo.com: a history of cybernetic animals and early robots [online]. [cit. 2017-07-07]. Dostupné z: http://cyberneticzoo.com/cyberneticanimals/1969-the-logo-turtle-seymour-papert-marvin-minsky-et-al-american/

[2] 2. Algoritmizace - Stránky k výuce informatiky [online]. [cit. 2018-07-13]. Dostupné z: http://www.ivt.mzf.cz/algoritmizace-a-programovani/uvod-do-algoritmu/2-algoritmizace/

[3] [online]. In: . [cit. 2017-07-07]. Dostupné z: http://old.spsemoh.cz/vyuka/algor/algritm.htm

[4] BELL, Tim, Ian H WITTEN, Mike FELLOWS, Robyn ADAMS a Jane MCKENZIE. Computer Science Unplugged: An enrichment and extension programme for primary-aged children [online]. 2006. Dostupné z: *http://csunplugged.org/*

[5] FILIP, Brož. Ozobot Evo je mnohem lepší než předchůdci, umí také komunikovat [online]. In: . [cit. 2017-07-07]. Dostupné z: http://jablickar.cz/ozobot-evo-je-mnohem-lepsi-nez-predchudci-umi-take-komunikovat/

[6] HAVELKOVÁ, Hana. [online]. In: . [cit. 2017-07-07]. Dostupné z: *http://wvc.pf.jcu.cz/ki/data/files/93prg1.ppt*

[7] HOFERKOVÁ, Kateřina. [online]. In: . [cit. 2017-07-07]. Dostupné z: *https://www.svethardware.cz/hra-cubetto-nauci-programovat-i-ty-nejmensi/42049*

[8] LESSNER, Daniel. HOW DO WE TRANSLATE COMPUTATIONAL THINKING INTO CZECH? [online]. In: . str. 1 [cit. 2018-04-20]. Dostupné z: http://ksvi.mff.cuni.cz/~lessner/w/data/_uploaded/file/papers/2014_02_lessner_didactig.pdf

[9] LESSNER, Daniel a Jiří VANÍČEK. Bobřík učí informatiku. Matematika – fyzika – informatika [online]. Praha: Prometheus, 2013, vol. 22, no. 5, pp. 374–382. Dostupné z: http://mfi.upol.cz/index.php/mfi/article/view/92/ 105

[10] Malý programátor [online]. [cit. 2017-07-07]. Dostupné z: *https://www.malyprogramator.cz/*

[11] Operational Definition of Computational Thinking for K-12 Education [online]. B.m.: International Society for Technology in Education (ISTE) a Computer Science Teachers Association (CSTA). 2011. Dostupné z: http://csta.acm.org/Curriculum/sub/CurrFiles/Co mpThinkingFlyer.pdf

[12] PITNER, Tomáš. Výuka programování na základní a střední škole [online]. In: . [cit. 2017-07-07]. Dostupné z: http://www.fi.muni.cz/~tomp/semuc/text_pitner.html

[13] Primo [online]. [cit. 2017-07-07]. Dostupné z: https://www.primotoys.com/

[14] Propedeutika. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 25. 8. 2016 [cit. 2017-07-06]. Dostupné z: *https://cs.wikipedia.org/wiki/Propedeutika*

[15] SGP Baltík 3. SGP Systems: Baltík C# 3D vizuální výukové programovací nástroje pro děti, mládež a dospělé [online]. [cit. 2017-07-07]. Dostupné z: https://www.sgpsys.com/cz/product_B3.asp

[16] Startuje projekt PRIM [online]. [cit. 2018-04-20]. Dostupné z: *http://wvc.pf.jcu.cz/ki/?article=/aktuality/startuje-projekt-prim.html*

[17] Tvořivost. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-04-20]. Dostupné z: *https://cs.wikipedia.org/wiki/Tvořivost*

[18] WING, Jeannette M. Computational Thinking: What and Why? [online]. 2010 [cit. 2013-10-28]. Dostupné z: https://www.cs.cmu.edu/~CompThink/papers/Th eLinkWing.pdf

8 Seznam obrázků

9 Přílohy

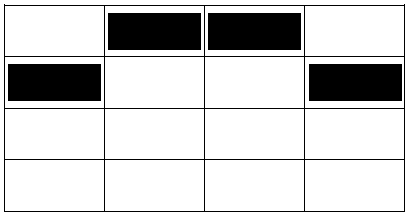
Příloha č. 1a – Sestavy a dohodnuté symboly

Příloha č. 1b – Ukázka možného řešení

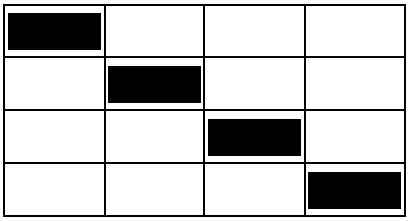
Příloha č. 1c – Prázdné předlohy (hrací kartičky)

Příloha č. 2 – Počáteční a cílová pozice kelímků

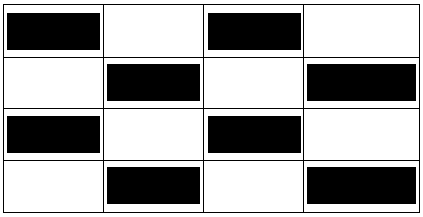
Příloha č. 1a - Sestavy a dohodnuté symboly



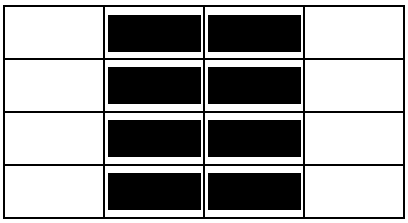
Pozice 1



Pozice 2



Pozice 3



Pozice 4

Dohodnuté symboly:

- ➔ o jedno místo doprava
- ➔ o jedno místo dolu
- ← o jedno místo poleva
- ● vyplnit barvou
- ↑ o jedno místo nahoru

Příloha č. 1b – Ukázka možných řešení

Součástí řešení je také rozšíření o cykly, kde je ovšem nutné předem dohodnout zápis.

Pro tento zápis jsem zvolil dvě varianty, a to variantu pro opakování jednoho příkazu, v tom případě se před příslušný symbol vloží číslovka, značící počet opakovaní a pro skupinu příkazů, kde je tato skupina ohraničena závorkami a opět se před závorku napíše počet opakování.

Pozice 1:

(startovní buňka je vlevo nahoře)

↓ o → ↑ o → o → ↓ o

Jelikož se určitá sekvence zápisu opakuje, lze toho využít pro zavedení opakování (cyklu)

↓ o → ↑ 2(o →) ↓ o

Pozice 2:

(startovní buňka je vlevo nahoře)

o → ↓ o → ↓ o → ↓ o

Jelikož se určitá sekvence zápisu opakuje, lze toho využít pro zavedení opakování (cyklu)

3(o → ↓) o

Pozice 3:

(startovní buňka je vlevo nahoře)

o → → o → ↓ o ← ← o ← ↓ o → → o → ↓ o ← ← o

Jelikož se určitá sekvence zápisu opakuje, lze toho využít pro zavedení opakování (cyklu)

2(o 2 → o → ↓ o 2 ← o ← ↓)

Pozice 4:

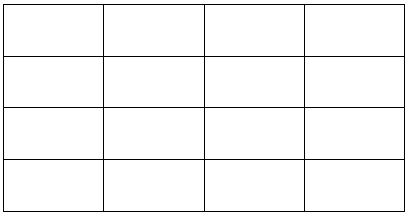
(startovní buňka je vlevo nahoře)

→ → o ↓ o ↓ o ↓ o → o ↑ o ↑ o ↑ o

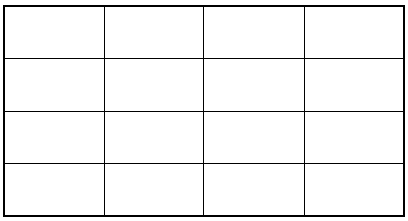
Jelikož se určitá sekvence zápisu opakuje, lze toho využít pro zavedení opakování (cyklu)

2 → 3(o ↓) o → 3(o ↑) o

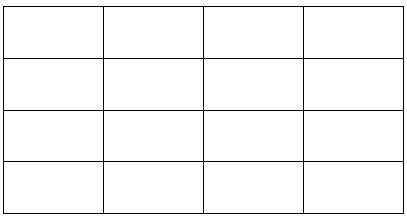
Příloha č. 1c – Prázné předlohy:



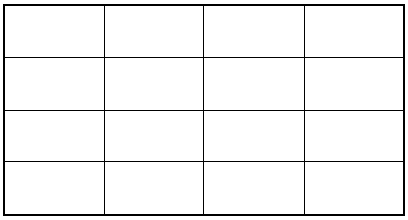
Pozice 1



Pozice 2



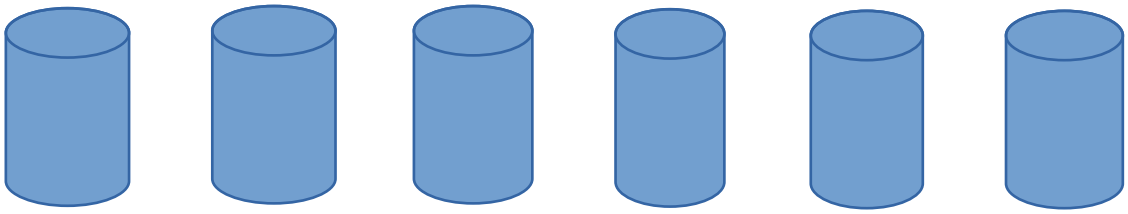
Pozice 3



Pozice 4

Příloha č.2 – Počáteční a cílová pozice kelímků

Počáteční pozice kelímků



Cílová pozice kelímků, kterou mají za úkol žáci dosáhnout.