Univerzita Karlova
Pedagogická fakulta
Katedra informačních technologií a technické výchovy

DIPLOMOVÁ PRÁCE

Utváření představ a osvojování vybraných konceptů používaných při sestavování programů s testovacími podmínkami na 1. stupni ZŠ

Examining the formation of ideas and learning about some programming concepts in the primary school

Bc. Radek Čuma

Vedoucí práce:	doc. RNDr. Miroslava Černochová, CSc.
Studijní program:	Učitelství pro střední školy (N7504)
Studijní obor:	Učitelství VVP pro ZŠ a SŠ – informační a komunikační technologie N IT (7504T276)

2018

Odevzdáním této diplomové práce na téma Utváření představ a osvojování vybraných konceptů používaných při sestavování programů s testovacími podmínkami na 1. stupni ZŠ potvrzuji, že jsem ji vypracoval pod vedením vedoucího práce samostatně za použití v práci uvedených pramenů a literatury. Dále potvrzuji, že tato práce nebyla využita k získání jiného nebo stejného titulu.

Praha, 12. 7. 2018

Rád bych poděkoval vedoucí diplomové práce doc. RNDr. Miroslavě Černochové za její cenné rady, ochotu a za trpělivost při vedení diplomové práce. Zároveň bych chtěl poděkovat své rodině, zvláště manželce, za podporu a trpělivost během celého studia a v neposlední řadě mojí ředitelce, která mě po celou dobu studia také podporovala a umožnila mi vytvořit podmínky pro realizaci navrženého pedagogického experimentu.

12. 7. 2018

……

Podpis

ABSTRAKT

Diplomová práce se zabývá představami žáků o smyslu a funkčním principu použití příkazů s testovacími podmínkami při sestavování algoritmů. Hlavním cílem diplomové práce je na základě teoretických východisek navrhnout metodické postupy pro výuku vybraných algoritmických konceptů na 1. stupni základní školy, naplánovat a realizovat pedagogický experiment a v jeho průběhu ověřovat funkčnost navržených metodických postupů. Ověřování funkčnosti navržených metodických postupů probíhalo na základě analýzy dat, které byly nashromážděné průběžným zapisováním poznatků o chování, pokroku a průběhu řešení vybraných úloh, pořizováním videozáznamů zachycujících průběh řešení jednotlivých úloh v rámci navržených „unplugged“ aktivit, pomocí nástroje pro monitorování pokroku žáků ve virtuálním programovacím prostředí Code.org, pořízením audionahrávek testování žáků pomocí řízeného rozhovoru a pomocí fotodokumentace tvorby vlastních bloků příkazů, ve kterých žáci přepisovali věty z mateřského jazyka do jazyka pro řízení počítače/stroje. Na základě syntézy získaných poznatků byly provedeny návrhy na úpravy navržených metodických postupů, aby se zjištěné obtíže, kterým žáci při výuce čelili, co nejvíce eliminovaly.

KLÍČOVÁ SLOVA

Testovací podmínky, algoritmizace, programování, informatické myšlení, unplugged aktivity, Code.org

ABSTRACT

This diploma thesis maps pupils' understanding about a functional principle of using commands along with testing conditions (IF, IF - THEN, REPEAT - UNTIL, etc.) when creating algorithms. The main aim of the thesis is to design and implement a set of lessons and a teaching approach based on a theory about learning of algorithmic concepts at primary education for pupils (aged in 9-11) with the intention of verifying a functionality of designed teaching procedures and their possible impacts on pupils' understanding. Data was collected through continuous monitoring of pupils' behavioural characteristics, progress and solution of chosen tasks, video recordings of task solving within the suggested unplugged activities, using a virtual tool Code.org for monitoring of a pupils' progress, audio recordings of interview with pupils, and photographs capturing a creation of own blocks of commands set up by a transcription from pupils´ mother language into a machine language (programming language) have all been used for a verification process of the designed teaching approach. By combining the acquired data sets, adjustments of these procedures have been made in order to eliminate the most frequent problems that pupils have encountered during teaching. The case study findings revealed that it is important for correct understanding of algorithmical procedures by primary school pupils to combine learning activities in a computer platform (like Code.org) with "unplugged" ones with material aids (a set of paper cards, etc.) and to take care of reading of algorithms and intrepreting them with using of another ones (lego cars, toys, etc.).

KEYWORDS

Testing conditions, programming cycles, algorithm design, programming, computational thinking, unplugged activities, Code.org

Obsah

1 Úvod

„Mladí lidé se dnes dělí na dvě skupiny: Jedni tráví svůj volný čas u hracích automatů a do omrzení hrají počítačové hry, druzí se baví programováním. Za deset, patnáct let budou ti druzí šéfy těch prvních. Teorie programování nás učí, jak metodicky správně analyzovat problém, navrhnout správný algoritmus a napsat přehledný, čitelný a strukturovaný program. Praxe nás učí, jak se vyrovnat s tím, že nic z toho pořádně neumíme. Programování není žádná idylka - je to boj. Boj s problémem, překladačem, operačním systémem, počítačem, nečekanými situacemi, které se nenajdou v učebnicích, a především s vlastní hloupostí, neznalostí a netrpělivostí."

Ivan Kopeček, Jan Kučera: Programátorské poklesky (Mladá fronta, Praha 1989)

V životě lidí dnešní informační společnosti čím dál více hrají významnou roli počítačové (digitální) technologie. Uživatelé počítačových technologií mohou v současnosti pracovat s technologiemi připojenými k internetu a používat nejrůznější aplikace, aniž do detailů vědí, jak tyto technologie fungují. Pryč je doba, kdy se programátor dorozumíval s počítačem skrze textové rozhraní a uživatelé počítačů byli spíše jejich nadšenci. Digitální technologie se staly nedílnou součástí každodenního života lidí na celém světě. Jejich rozvoj mění zásadním způsobem náš život, naši práci, naše učení.

Dovednosti spojené s obsluhou a ovládáním digitálních technologií se staly významným artiklem na trhu práce. Česká republika patří mezi vyspělé země Evropské unie a kvůli zvyšující se globální konkurenci je stále obtížnější zajišťovat soustavný ekonomický růst, zamezovat zvyšování nezaměstnanosti a tím pádem zajišťovat vysokou životní úroveň a spokojený život všem občanům. Svět, ve kterém žijeme, a budou žít následující generace, se díky vlivu digitálních technologií zásadně mění a v souvislosti s těmito změnami musí zákonitě dojít také ke změnám ve vzdělávacím procesu. Základní úroveň digitálních dovedností vyžaduje více než 90 % profesí.[1]Bez počítačů si nedovedeme představit různá

[1] EURACTIV.CZ. Programování – Je Česká republika připravená?. [online] 2015. Dostupné z: http://euractiv.cz/factsheet/obchod-a-export/programovani-je-ceska-republika-pripravena-000136/

odvětví všech sektorů hospodářství. Vzhledem k tomuto faktu je nezbytně nutné, aby škola žáky naučila digitální technologie účelně a efektivně používat, a to zejména při řešení problémů každodenního života a různých úloh. Některé typy úloh a reálných problémů je mnohem efektivnější řešit pomocí počítače a některé typy úloh se dokonce bez počítače řešit nedají, anebo jen velmi obtížně. I přes to, že se v počítačích a počítačovém zpracování informací skrývá velký edukační potenciál a naprostá většina lidí si v dnešní době nedokáže život bez ICT připojených k internetu představit, je využití počítačů zejména ve vzdělávacím procesu a ke vzdělávacím účelům v naší společnosti stále velmi nedostatečné.

Dle průzkumů ČŠI vyplývá[2], že naprostá většina žáků základních škol tráví denně nějaký čas s ICT, avšak jen minimum z nich využívá tento čas ke vzdělávání nebo pro přípravu na výuku. Proto by mělo být úkolem školy naučit žáky tyto technologie a jejich potenciál využívat, zejména pak právě ke vzdělávacím účelům. Povinná výuka ICT předmětů se na školy dostala až na přelomu tisíciletí, avšak vzhledem k masivnímu šíření a rozvoji výukových programů a jiných uživatelských aplikací se výuka na našich školách spíše zaměřuje na rozvoj uživatelských dovednosti žáků bez hlubšího pochopení a porozumění, jak počítač vůbec funguje. Uživatelské dovednosti si žáci díky neustálému rozvoji a každodennímu používání různých ICT vytvářejí víceméně sami a tudíž není nutné jejich rozvoj preferovat při výuce. Dnes není za digitálně gramotného člověka považován ten, který dokáže ICT obsluhovat, ale spíše ten, který má alespoň základy informatického myšlení, rozumí základům programování, ví, jak digitální technologie fungují, a dokáže účelně a efektivně využívat digitální technologie při řešení problémů, pro svůj osobní rozvoj a k občanským aktivitám[3].

Proto je v dnešním světě, ve kterém jsme stále více technologiemi ovlivňováni, důležité již od útlého věku tyto informatické dovednosti rozvíjet. Mezi informatické dovednosti bezesporu patří také programování a algoritmické myšlení, které zároveň napomáhá rozvoji tzv. informatického myšlení, jakožto klíčové gramotnosti pro 21. století.

---

[2] ČESKÁ ŠKOLNÍ INSPEKCE. Hlavní zjištění ICILS 2013. [online] 2014 [cit 2017-09-24]. Dostupné z: http://www.csicr.cz/Prave-menu/Mezinarodni-setreni/ICILS/Ceska-skolni-inspekce-zverejnuje-vysledky-setreni

[3] NEUMAJER, O. Konference Počítač ve škole 2017: Být digitálně gramotný už neznamená jen ovládat počítač. [online] 27. 4. 2017 [cit. 2017-07-28]. Dostupné z: http://ondrej.neumajer.cz/konference-pocitac-ve-skole-2017-byt-digitalne-gramotny-uz-neznamena-jen-ovladat-pocitac/

Tato situace se ovšem nedá změnit kvůli stávajícímu pojetí našich kurikulárních dokumentů, které jsou odbornou veřejností považovány za příliš obecné, zastaralé, bez jasné koncepce, nereflektující vývoj v dané oblasti a nové požadavky na rozvoj informačně technologických kompetencí žáků. Tyto dokumenty je podle odborníků ve vzdělávání ICT nutné co nejdříve přepracovat z hlediska obsahu i hodinové dotace ICT předmětů a z hlediska jejich integrace do ostatních vyučovacích předmětů[4]. V blízké budoucnosti uvidíme, jestli připravovaná Strategie digitálního vzdělávání do roku 2020 a následná aktualizace a modernizace RVP pomůže tyto zásadní nedostatky našich stávajících kurikulárních dokumentů odstranit.

Téma své diplomové práce jsem si vybral záměrně a vedlo mě k tomu více okolností:

- připravované změny RVP s ohledem na Strategii digitálního vzdělávání do roku 2020,
- testovací podmínky (Opakuj Xkrát, Opakuj – Dokud, Pokud, Pokud - Jinak) patří mezi základní algoritmické konstrukce,
- skutečnost, že děti baví aktivity s počítači,
- moje přesvědčení, že je zapotřebí učit děti přemýšlet,
- moje přesvědčení, že by žáci při učení měli prožívat radost.

Ministerstvo školství v současné době aktualizaci RVP připravuje a algoritmizaci či programování budou přepracované RVP nově obsahovat[5]. Připravované změny RVP se pak zákonitě budou muset projevit v úpravách ŠVP jednotlivých škol, které budou muset začlenit výuku algoritmizace a programování již na 1. stupeň ZŠ. Mezi hlavní cíle Strategie digitálního vzdělávání do roku 2020[6] patří mimo jiné rozvíjení informatického myšlení žáků, které se dá charakterizovat jako schopnost rozložit řešení problémů nebo úloh do několika jednodušších dílčích kroků a zároveň najít a zobecnit vzorce těchto řešení. Algoritmické myšlení pak dokáže pomoci rychleji a efektivněji řešit běžné problémy i z reálného života.

[4] RAMBOUSEK, V. studijní text z předmětu Edukační technologie. 2014 [cit 2017-09-20]. Dostupné z: http://moodle.it.pedf.cuni.cz/course/view.php?id=1114

[5] NEUMAJER, O. RVP jsou zastaralé. Změnilo se toho tolik, že nemohou odpovídat potřebám digitálních kompetencí. [online] 11. 3. 2017 [cit. 2017-07-28]. Dostupné z: http://www.ceskaskola.cz/2017/03/ondrej-neumajer-rvp-jsou-zastarale.html

[6] MŠMT. Strategie digitálního vzdělávání do roku 2020. [online] 31. 10. 2014. [cit. 2017-07-28]. Dostupné z: http://www.msmt.cz/vzdelavani/skolstvi-v-cr/strategie-digitalniho-vzdelavani-do-roku-2020

Testovací podmínky a opakování (*Opakuj Xkrát, Opakuj – Dokud, Pokud, Pokud - Jinak*) patří mezi základní algoritmické konstrukce. Schopnost objevit opakující se činnosti a dokázat je zefektivnit se hodí každému člověku v běžném životě. Nejzásadnější změnou v myšlení dětí při výuce programování je totiž samotné řešení úkolu. Ve škole často bývají na všechno šablony. Tady je vzorec, sem se dosadí veličiny, tady je ukázkový příklad a jde se na další příklady. Pro programování je ale klíčovou dovedností schopnost pochopit problém, rozložit jej na menší dílčí kroky a vymyslet funkční (nejlépe optimální) řešení, což se následně odráží i v samotném životě. Ne všichni musí být jednou programátoři, ale samostatně myslící a logicky uvažující člověk by měl být každý.[7]

Jak už bylo řečeno, z průzkumů ČŠI vyplývá, že naprostá většina žáků základních škol tráví denně nějaký čas s ICT, stále platí, že ne všichni žáci mají mimo školu k těmto technologiím stejný přístup. Je nutné, aby se právě škola snažila tyto rozdíly vyrovnávat a nabízet rovné příležitosti k rozvoji digitální gramotnosti a informatického myšlení všem žákům způsobem odpovídajícím jejich individuálním možnostem a podmínkám. V tomto ohledu je jedním z hlavních úkolů dnešní školy a formální výuky takto vznikající digitální rozdíly překonávat. Ve světě, ve kterém jsme stále více obklopeni a ovlivňováni technologiemi, je důležité již od útlého věku žáků rozvíjet a napomáhat chápání způsobu „uvažování“, který používají počítače, což potvrzují také výzkumy, které se zaměřují na význam vzdělávání s využitím ICT prostředků v raném dětství na snížení tzv. sociální digitální propasti, která odděluje žáky z rodin s nízkými příjmy a jinak znevýhodněné žáky od jejich vrstevníků.[8] Avšak ani sebelepší kombinace připojení k internetu, softwaru a zařízení nenahradí učitele, kteří by měli vzdělávacím procesem žáky provázet.[9] V tomto ohledu je nutné, aby škola měla kvalitní, aprobované a odborně schopné učitele s odpovídajícími dovednostmi z hlediska metodiky výuky informačních předmětů.

---

[7] MALÝ, M. Jak naučit děti logickému myšlení a schopnosti řešit problémy? Jde to!. [online] 21. 07. 2017 [cit. 2017-09-03]. Dostupné z: https://www.lupa.cz/clanky/jak-naucit-deti-logickemu-mysleni-a-schopnosti-resit-problemy-jde-to/

[8] DAUGHERTY, L. Early Education Plays Role in Bridging the 'Digital Divide'. RAND Corporation [online]. 2014. Dostupné z: http://www.rand.org/news/press/2014/03/03.html

[9] DAUGHERTY, L., DOSSANI, R., JOHNSON, E. E., OGUZ, M. Using Early Childhood Education to Bridge the Digital Divide. RAND Corporation [online]. 2014. Dostupné z: http://www.rand.org/pubs/perspectives/PE119.html

Díky své dlouholeté praxi vím, že žáky výuka programování baví. Žáky pak při práci motivuje fakt, že mohou vidět výsledek programování téměř ihned, přičemž se učí přemýšlet a myslet informaticky. Jelikož mimo jiné také vyučuji informatické předměty na 1. stupni ZŠ, vedla mě v neposlední řadě k výběru tématu mé diplomové práce touha po zlepšení mých učitelských vědomostí a dovedností v oblasti výuky programování a touha dokázat zjistit, které věci dělají dětem největší potíže a co je pro ně z hlediska chápání základních algoritmických konstrukcí nejtěžší. Věřím, že moje práce může posloužit i jako inspirace a metodický materiál pro skupinu neaprobovaných učitelů bez odborné způsobilosti, kteří vyučují informatické předměty na 1. stupni ZŠ, a ukázat jim, jakým způsobem lze efektivně vyučovat základy programování a rozvíjet algoritmické myšlení žáků.

2 Vymezení výzkumného problému

Hlavním výzkumným problémem diplomové práce je ověřování funkčnosti navržených a realizovaných metodických přístupů, které jsou založené na aplikování činností bez počítače (dále jen „unplugged“ aktivity), a na ně navazujících aktivit ve vybraném virtuálním programovacím prostředí, při výuce vybraných algoritmických konceptů (*Opakuj Xkrát, Opakuj – Dokud, Pokud, Pokud - Jinak*) na 1. stupni základní školy. Diplomová práce bude na základě nashromážděných dat následně zkoumat, zda žáci při výuce správně porozuměli a osvojili si vybrané algoritmické koncepty, a identifikovat problémy, kterým žáci při osvojování těchto algoritmických konceptů čelili.

Informatické předměty na 1. stupni základních škol obvykle vyučují učitelé aprobovaní na všeobecnou výuku žáků prvního stupně bez hlubších znalostí a dovedností v oblasti ICT. Vzhledem k tomu se naprostá většina žáků 1. stupně dnešních základních škol s výukou programování a algoritmického myšlení nesetkává, anebo jen povrchně. I proto je snahou diplomové práce ukázat, jakými způsoby, metodickými přístupy a vyučovacími metodami lze vyučovat základy programování a algoritmického myšlení na 1. stupni základní školy, aby byla výuka pro žáky zajímavá, zábavná, motivující, a zároveň aby žáci dokázali správně porozumět, osvojit si a automaticky používat vybrané algoritmické koncepty při sestavování počítačových programů.

Diplomová práce se zabývá metodickými aspekty výuky základů programování s využitím výukových metod, které se opírají o „unplugged“ aktivity a na ně navazující aktivity ve vybraném programovacím prostředí. Takovéto aktivity pomáhají malým dětem, ale i začátečníkům, s propojením reálného a hmotného světa s virtuálním světem počítačového programování. Východiskem pro vymezení výzkumného problému je také snaha zjistit a analyzovat možné problémy, kterým žáci budou při výuce čelit. Okrajově se diplomová práce zabývá otázkou, které virtuální programovací prostředí jsou vhodné pro výuku základů programování a také možností využití edukační robotiky ve výuce základů programování na 1. stupni ZŠ.

3 Cíl a metodika práce

V této části bude vytyčen hlavní cíl diplomové práce, od kterého se pak budou odvíjet jednotlivé dílčí cíle práce. S ohledem na jednotlivé dílčí cíle práce budou stanoveny konkrétní úkoly a popsány použité metody.

3.1 Hlavní cíl práce

Hlavním cílem diplomové práce bylo na základě teoretických východisek navrhnout, realizovat a prakticky pomocí pedagogického experimentu ověřit metodické postupy pro výuku zaměřenou na utváření dovedností sestavovat algoritmy s testovacími podmínkami (*Opakuj Xkrát, Opakuj – Dokud, Pokud, Pokud – Jinak*) na 1. stupni základní školy, které jsou založené na „unplugged“ aktivitách a na ně navazujících aktivitách ve vybraném virtuálním programovacím prostředí.

3.2 Dílčí cíle práce a úkoly

Pro splnění hlavního cíle byly vymezeny následující dílčí cíle a z těchto cílů vyplývající jednotlivé dílčí úkoly.

Cíl 1: Zmapovat dosud existující postupy ve výuce algoritmizace s využitím testovacích podmínek.

> **Úkol 1:** Vyhledat a seznámit se s dosud existujícími postupy ve výuce základních algoritmických konceptů, zejména pak ve výuce testovacích podmínek.

Cíl 2: Navrhnout „unplugged“ aktivity, které vedou k porozumění, osvojení a používání vybraných algoritmických konceptů, zejména pak testovacích podmínek, a které mají za úkol snadnější propojení reálného světa s virtuálním světem programovacího prostředí u žáků 1. stupně základních škol.

> **Úkol 2:** Analyzovat dostupné úlohy a didaktické materiály, které mohou přispět k rozvíjení algoritmických dovedností žáků 1. st. ZŠ spojených navrhováním algoritmů s použitím testovacích podmínek. Jedná se o úlohy v informatické soutěži Bobřík informatiky, v matematické soutěži Matematický klokan, v bezplatné online aplikaci pro výuku programování Code.org, základní lekce pro výuku programování

malého programovatelného robota Ozobot a jiných dostupných publikovaných úloh v učebnicích nebo na internetu.

Úkol 3: Na základě již ověřených postupů ve výuce testovacích podmínek navrhnout „unplugged“ aktivity, které vedou k porozumění, osvojení a používání základních a klíčových algoritmických konceptů, zejména pak testovacích podmínek, a které mají za úkol snadnější propojení reálného a hmotného světa s virtuálním světem programovacího prostředí u žáků 1. stupně základních škol.

Úkol 4: Připravit materiální didaktické prostředky, které jsou potřebné pro realizaci navržených „unplugged“ aktivit ve výuce.

Cíl 3: Navrhnout úlohy a aktivity pro žáky 1. stupně ZŠ s použitím vhodných volně dostupných počítačových aplikací a edukačních programovacích jazyků, pomocí kterých žáci porozumí a osvojí si použití testovacích podmínek při sestavování počítačových programů.

Úkol 5: Zmapovat volně dostupné počítačové aplikace a edukační programovací jazyky vhodné pro výuku základů programování na 1. stupni základních škol, pomocí kterých si žáci mohou osvojovat používání testovacích podmínek při sestavování počítačových programů.

Úkol 6: Navrhnout úlohy a aktivity pro výuku testovacích podmínek žáků 1. stupně ZŠ ve vybrané počítačové aplikaci, pomocí kterých si žáci budou osvojovat používání testovacích podmínek při sestavování počítačových programů.

Cíl 4: Naplánovat a prakticky realizovat navržené metodické postupy, které jsou založené na navržených „unplugged“ aktivitách a aktivitách ve vybrané počítačové aplikaci.

Úkol 7: Navrhnout a naplánovat výuku testovacích podmínek pomocí navržených „unplugged“ aktivit a na ně navazujících úloh a aktivit ve vhodné počítačové aplikaci.

Úkol 8: Realizovat navrženou výuku zaměřenou na testovací podmínky pomocí navržených „unplugged“ aktivit a navržených úloh a aktivit ve vybrané počítačové aplikaci v rámci výuky informatického předmětu 3. - 5. ročníku ZŠ.

Úkol 9: Průběžně testovat porozumění žáků vybraným algoritmickým konceptům a na základě výsledků testování jim poskytovat průběžnou zpětnou vazbu.

Cíl 5: Získat a nashromáždit výzkumná data, pomocí kterých se bude následně ověřovat a vyhodnocovat porozumění žáků vybraným algoritmickým konceptům a analyzovat obtíže, kterým žáci při výuce, založené na používání „unplugged“ aktivit, čelili.

Úkol 10: Průběžně monitorovat a zaznamenávat chování, pokroky a průběh řešení vybraných úloh žáky během výuky, a tím získávat data o jejich práci.

Úkol 11: Pořídit audio záznamy ústních výstupů žáků, týkající se jejich porozumění a osvojování vybraných algoritmických konstrukcí.

Úkol 12: Nashromáždit výstupy žáků z jednotlivých činností. Jedná se o obrazovou dokumentaci výsledků práce s materiálními didaktickými prostředky v průběhu „unplugged“ aktivit a soubory s výsledky práce ve vybraném volně dostupném programovacím prostředí.

Cíl 6: Vyhodnotit získané zkušenosti a výzkumná data, týkající se testovacích podmínek s použitím „unplugged“ aktivit a úloh a aktivit ve vybrané počítačové aplikaci.

Úkol 13: Analyzovat nashromážděná data z jednotlivých činností v průběhu „unplugged“ aktivit a aktivit ve vybraném volně dostupném programovacím prostředí.

Úkol 14: Analyzovat audio záznamy ústních výstupů žáků, týkající se jejich porozumění a osvojování vybraných algoritmických konceptů.

Úkol 15: Identifikovat obtíže, kterým žáci čelili během výuky.

Úkol 16: Modifikovat navržené a realizované metodické přístupy tak, aby se problémy žáků při osvojování vybraných algoritmických konstrukcí co nejvíce eliminovaly.

3.3 Použité metody

V teoretické části diplomové práce budou za účelem zmapování dosud existujících postupů ve výuce vybraných algoritmických konceptů a zmapování vhodných volně dostupných počítačových aplikací a edukačních programovacích jazyků pro výuku základů programování na 1. stupni základních škol použity teoretické metody, založené na vyhledávání, analýze, porovnání a syntéze dostupných zdrojů informací. Za účelem návrhu „unplugged" aktivit a aktivit ve vybraném virtuálním programovacím prostředí budou použity teoretické metody, založené na syntéze předchozích poznatků a teoretických východisek dosud existujících metodických přístupů k výuce vybraných algoritmických konceptů. Empirickou metodou bude ve výzkumné části diplomové práce metoda akčního výzkumu zahrnující plánování, činnost, pozorování, reflexi a nové plánování (McNiff, 1988).[10] V případě řešení diplomového úkolu to znamená:

Tabulka č. 1 – Etapy akčního výzkumu v případě řešení diplomového úkolu

Plánování	Návrh úvodních motivačních aktivit s Ozobotem, návrh „unplugged" aktivit, návrh úloh a aktivit ve virtuálním programovacím prostředí, návrh pedagogického experimentu
Činnost	Realizace pedagogického experimentu
Pozorování	Monitorování činností, výsledků a pokroku žáků při řešení jednotlivých úloh
Reflexe	Podrobná analýza nashromážděných dat
Nové plánování	Návrh na úpravy a změny

Výzkumná data se budou shromažďovat průběžným zapisováním poznatků o chování, pokroku a průběhu řešení vybraných úloh, pořizováním videozáznamů zachycujících průběh řešení jednotlivých úloh v rámci navržených „unplugged" aktivit, pomocí nástroje pro monitorování pokroku žáků ve virtuálním programovacím prostředí Code.org, pořízením

[10] McNiff, J. Action research: Principles and Practice. London: Macmillan Education, 1988. Citováno v NEZVALOVÁ, D. Akčním výzkumem k zlepšení kvality školy. e-Pedagogium [online]. 2002, roč. 2, č. 4. [cit. 2017-10-22]. Dostupné z: http://epedagog.upol.cz/eped4.2002/clanek02.htm

audionahrávek testování žáků pomocí řízeného rozhovoru a pomocí fotodokumentace tvorby vlastních bloků příkazů, ve kterých žáci přepisovali věty z mateřského jazyka do řeči srozumitelné pro stroje, do programovacího jazyka. Nashromážděná data budou podrobně analyzována, vyhodnocena a pomocí syntézy získaných poznatků budou provedeny návrhy na úpravy realizovaných a zkoumaných metodických postupů, aby se zjištěné problémy žáků při výuce vybraných algoritmických konceptů co nejvíce eliminovaly.

4 Teoretická část

Při navrhování a sestavování počítačového programu používají žáci různé algoritmické koncepty a příkazy. Ke stejnému výsledku mohou žáci dojít rozdílnou cestou a tempem.

Programování se nelze naučit na základě memorování. Výuka základům programování a sestavování počítačových programů a algoritmizace úloh je založena na aktivním přemýšlení. Při programování se uplatňují dovednosti algoritmicky myslet, mít schopnost abstrakce, umět přemýšlet o automatizaci procesů, zvládat práci s formálním/i jazykem/jazyky a uvědomit si meze praktické vyčíslitelnosti (limits of practical computability, viz Hromkovič et al., 2016, s. 112-114). „Tvorba algoritmu se rozvíjí v kontextu prostřednictvím procesů, jako je pokus a omyl, hra, spolupráce, diskuse a vyjednávání“ (Kitchin, 2017, s. 18).

Tím, že umožníme žákům, aby sestavovali počítačové programy pro zajímavé hravé aktivity, se naplňuje celoživotní idea Seymoura Paperta z MIT, který už na začátku 60. let 20. století byl přesvědčen, že by se děti měly učit sdělovat počítači, aby dělal to, co děti vymyslí. Seymour Papert ve své knize *MINDSTORMS. Children, Computers, and Powerful Ideas* ukazuje, jak mohou počítače působit na to, jak člověk myslí a jak se učí (Papert, 1980, s. 3). Aktivní a tvořivá činnost na počítači může dětem přinést silné myšlenky, může jim pomoci objevovat nové souvislosti mezi poznatky. Důležité však je, aby děti o svých postupech povídaly, aby je verbalizovaly a na druhou stranu aby své náměty, nápady formalizovaly do abstraktnějších schémat, do jazyka stroje, robota. Papert se inspiroval dílem Jeana Piageta, jeho „modelem dětí jako budovatelů svých vlastních intelektuálních struktur“ (Papert, 1980, s. 7). Papert byl přesvědčen, že „děti se mohou naučit používat počítač mistrovským způsobem a že to, jakým způsobem se učí používat počítač, může ovlivnit a změnit i to, jak se učí i jiným věcem“ (Papert, 1980, s. 8).

Papert se vždy zajímal o to, „jak se žáci zapojují do konverzace (ať už sami se sebou nebo s jinými žáky) se svými produkty/artefakty. Tyto rozhovory mají velký význam v procesu učení člověka, usnadňují osvojování nových poznatků. Papert vždy kladl důraz na význam nástrojů, médií a kontextů v rozvoji člověka“ (Ackermann, 2001, s. 1).

Samotné sestavování algoritmů a programování počítače může žáky zaujmout už tím, že pracují na něčem, co má pro ně nějaký význam. Většina žáků tráví denně hodně času s technologiemi. Nejvíce času tráví komunikací na sociálních sítích a hraním počítačových her. Jen někteří žáci přemýšlejí, jak by se daly tyto technologie ovládat, že počítačová hra je vlastně funkční počítačový program, který musel někdo sestavit. Robotizace a automatizace nás bude čím dál více obklopovat v běžném životě, a tudíž i poptávka po schopných programátorech, kteří budou umět roboty naprogramovat a ovládat, bude logicky úměrně s používáním robotů a automatů stoupat.[11]

MŠMT v současné době připravuje aktualizaci RVP; revidované RVP budou klást důraz na rozvoj digitální gramotnosti napříč všemi vzdělávacími oblastmi a místo vzdělávací oblasti ICT bude do kurikula zařazena Informatika, která by měla přispět k rozvoji informatického myšlení žáků. Základní školy budou muset do svých ŠVP začlenit výuku programování již na 1. stupeň.

Jakým způsobem z metodologického hlediska začlenit výuku algoritmizace do školního vzdělávání na ZŠ? Pitner (2000) navrhl dva přístupy k výuce algoritmizace: strukturovaný a objektově orientovaný (Broumová, 2012, s. 14). Podle Kopecké (1998, zmíněno v Pitner, 2000) jsou žáci 4. - 5. ročníků ZŠ schopni chápat algoritmické postupy v podobě hotových programů, jednoduchých příkazů a sekvenci příkazů a žáci 6. - 7. ročníků už v podobě větvení programu (1 podmínka), s jedním cyklem a proměnnými.

Testovací podmínky (Opakuj Xkrát, Opakuj – Dokud, Pokud, Pokud – Jinak) patří mezi základní a klíčové algoritmické koncepty, se kterými by se žáci při výuce základů programování a algoritmického myšlení měli seznámit a následně je umět používat při sestavování programů. Edukační programování umožňuje dětem přistupovat k počítačům relativně jednoduchou, zábavnou a navíc tvůrčí formou. Výsledky programování mohou žáci vidět okamžitě: program buďto funguje, nebo nevykonává to, co si autor (žák, žáci) přál. Taková výuka pak dokáže děti a začátečníky dále motivovat k rozvíjení jejich dalších programátorských dovedností a zároveň s tím rozvíjet kreativní, logické, kritické, informatické a algoritmické myšlení.

[11] HLAVENKA, J. Když práce mizí aneb Ničí Internet střední třídu? [online] 7. 6. 2013. Dostupné z: https://www.lupa.cz/clanky/jiri-hlavenka-kdyz-prace-mizi-aneb-nici-internet-stredni-tridu/

4.1 Teoretická východiska práce

4.1.1 Výuka základních algoritmických konceptů v ČR a v zahraničí

Algoritmické myšlení se skládá z mnoha schopností a je také ovlivněno řadou kognitivních faktorů, jako je například abstraktní a logické myšlení, myšlení ve strukturách, kreativita nebo kompetence pro řešení problémů. Tato složitost způsobuje, že není jednoduché se algoritmické myšlení naučit, a vysvětluje potřebu dobrého didaktického přístupu zvláště pro začátečníky (Futschek, 2010).

Různí autoři vymezují různými způsoby, co se rozumí algoritmickým myšlením. Někteří autoři je dokonce považují za nejdůležitější kompetenci v souvislosti s výukou informatiky. „K algoritmickému myšlení lze přistupovat jako k určitému souboru schopností, které jsou spojeny s konstrukcí a pochopením algoritmů, mezi něž patří schopnost analyzovat daný problém, schopnost přesně vymezit problém, schopnost najít základní akce, které odpovídají danému problému, schopnost sestavit správný algoritmus k danému problému pomocí základních akcí, schopnost přemýšlet o všech možných zvláštních a normálních případech problému a schopnost navrhnout / vylepšit efektivnější algoritmus“ (Futschek, 2006, s. 160).

V ČR je otázka, jakými metodickými postupy učit malé děti programovat, poměrně nová, ale vzhledem k již zmiňovaným připravovaným změnám RVP velice důležitá; témata algoritmizace a programování budou muset být ve ŠVP jednotlivých škol jasně vymezena.

Bohaté zkušenosti se zaváděním programování do školního vzdělávání mají například Slovensko, Polsko, Anglie, Rusko.

Slovensko

Na Slovensku jsou základy informatiky součástí povinné vzdělávací oblasti Matematika a práca s informáciami, která se realizuje jako dva předměty Matematika a Informatika; velký důraz se klade na integraci algoritmického myšlení do moderního vzdělávání, jehož neoddělitelnou součástí je i edukační programování.

V učivu Informatiky pro 1. stupeň ZŠ (ISCED 1) se podle nových kurikulárních dokumentů prolínají dvě složky – jedna zaměřená na digitální technologie a druhá na budování základů informatiky (ŠPÚ, Informatika – primárne vzdelávanie, s. 2) tak, aby žáci dokázali uvažovat o algoritmech, hledali a nacházeli algoritmická řešení problémů, vytvářeli návody

a programy podle daných pravidel (ŠPÚ, Informatika – primárne vzdelávanie, s. 2). Problematika algoritmizace je zařazena do pěti tematických celků:

- *Algoritmické riešenie problémov – analýza problému*
- *Algoritmické riešenie problémov – interaktivne zostavovanie riešenia:* žáci na konci 4. ročníku by měli být schopni řešit problémy přímým řízením vykonavatele (robota, želvy, …) a aplikovat elementární příkazy daného jazyka pro řízení vykonavatele;
- *Algoritmické riešenie problémov – pomocou postupnosti príkazov:* žáci na konci 4. ročníku (ŠPÚ, Informatika – primárne vzdelávanie, s. 7) by měli být schopni řešit problém skládání příkazů do posloupnosti, dále doplnit, dokončit a modifikovat rozpracované řešení, interpretovat posloupnost příkazů, objevit chybu v posloupnosti příkazů;
- *Algoritmické riešenie problémov – interpretácia zápisu riešenia:* žáci na konci 4. ročníku ZŠ by měli umět dokázat (ŠPÚ, Informatika – primárne vzdelávanie, s. 7) realizovat návod, postup, algoritmus řešení problémů, interpretovat ho, krokovat řešení, simulovat činnost vykonavatele. V tematickém celku
- *Algoritmické riešenie problémov – hľadanie, opravovanie chýb:* žáci na konci 4. ročníku ZŠ by měli umět dokázat (ŠPÚ, Informatika – primárne vzdelávanie, s. 8) najít chybu po provedení algoritmu, najít a opravit chybu v návodě/ v zápise řešení a diskutovat o svých řešeních.

V učivu informatiky pro 2. stupeň ZŠ (ISCED 2) se podle nových kurikulárních dokumentů prolínají dvě složky – jedna zaměřená na digitální technologie a druhá na budování základů informatiky (ŠPÚ, Informatika – nižšie stredné vzdelavanie, s. 2) tak, aby žáci dokázali uvažovat o algoritmech, hledali a nacházeli algoritmická řešení problémů, vytvářeli návody a programy podle daných pravidel (ŠPÚ, Informatika – nižšie stredné vzdelavanie, s. 2). Problematika algoritmizace je zařazena do šesti tematických celků:

- *Algoritmické riešenie problémov – analýza problému*
- *Algoritmické riešenie problémov – jazyk na zápis riešenia*
- *Algoritmické riešenie problémov – pomocou postupnosti príkazov*
- *Algoritmické riešenie problémov – pomocou cyklov*
- *Algoritmické riešenie problémov – interpretácia zápisu riešenia*
- *Algoritmické riešenie problémov – hľadanie, opravovanie chýb*

Do nově zaváděného učiva informatiky pro gymnázia na Slovensku je zařazen tematický celek *Algoritmické rešenie úloh*, který tvoří společně s dalšími celky Reprezentácie a nástroje, Komunikácia a spolupráca, Softvér a hardvér a Informačná společnost obsahovou náplň předmětu Informatiky (ŠPÚ, Informatika – gymnázium, s. 1). Mezi výkonovými standardy tohoto kurikula je i požadavek na to, že žák dokáže „formulovat a neformálně (přirozeným jazykem) vyjádřit ideu řešení“ a že dokáže „plánovat řešení úlohy jako posloupnost příkazů větvení a opakování“ (ŠPÚ, Informatika – gymnázium, s. 9).

Ještě nedávno se učivo informatiky na ZŠ a SŠ na Slovensku členilo do pěti tematických okruhů, jedním z nich byl i okruh *Procedury, řešení problémů a algoritmické myšlení*. Tyto tematické jednotky byly zařazeny do všech úrovní vzdělávání, na každé úrovni však s jiným cílem:

- na úrovni ISCED 1 se věnovala pozornost rozvoji schopnosti žáků hledat řešení problémových úloh a ověřovat je s použitím ICT a schopnosti přesně vyjadřovat myšlenky a postupy a dokázat je formálně zapisovat (MŠ Slovenskej republiky, 2009, s. 11),
- na úrovni ISCED 2 kontroly pohybu nějakého objektu v dané oblasti,
- na úrovni ISCED 3 žáci pracují s efektivitou určitého algoritmu a pracují s opravdovým vyšším programovacím jazykem.[12]

Polsko

Žáci v Polsku mají samostatný předmět informatika v každém ročníku od prvního ročníku do maturity bez přerušení již od roku 1985. Od září 2015 bylo zavedeno kurikulum, podle něhož se každý žák v průběhu svého školního vzdělávání s programováním setká. Tento předmět je určen všem žákům. Programování ve výuce je chápáno jako nástroj k rozvoji (informatického) myšlení a k řešení problémů, nikoliv jako cíl naučit žáky pouze programovat.[13]

[12] BLAHO, A., SALANCI , L. Informatics in Primary School: Principles and Experience. In Kalaš, I. and Mittermeir, R. T. (eds.) ISSEP 2011, s. 129–142 (2011).

[13] LESSNER, Daniel. Střípky z konference Didinfo 2015 (2): Výuka informatiky v Polsku. Učíme informatiku [online]. 2015, (1) [cit. 2015-06-26]. Dostupné z: http://ucimeinformatiku.blogspot.cz/2015/06/stripky-z-konference-didinfo-2015-2.html

Itálie

„Unplugged“ aktivity mají pozitivní vliv na utváření představ o základních algoritmických konceptech, což dokládá výzkum university v italském Palermu, ve kterém se porovnávají dva odlišné přístupy k výuce algoritmického a informatického myšlení.[14] Výzkumu se zúčastnilo 251 žáků 3. a 4. ročníků základní školy a zabýval se otázkou, jestli je lepší přístup k výuce základních algoritmických konceptů metodou „unplugged“ aktivit a až v návaznosti na tyto aktivity programovat ve Scratch (132 žáků), nebo jestli je lepší začínat výuku základů programování ihned pomocí Scratch (119 žáků). Úlohy, které žáci řešili ve Scratch, byly zaměřeny na sekvenci základních příkazů, použití složitějších příkazů s testovacími podmínkami a cyklů při sestavování algoritmů. Žáci, kteří se učili metodou „unplugged“ aktivit, řešili scénář na sebe navazujících úloh zaměřených na provádění instrukcí, kdy jeden žák plnil instrukce, které mu zadával jiný žák, aniž by na sebe viděli. Instrukce byly dávány slovně a následně je žáci také zapisovali pomocí šipek. Cílem těchto „unplugged“ aktivit bylo, aby žáci porozuměli a pochopili, jak počítače přenášejí informace. Závěry výzkumu nepoukazují na to, že by byl jeden metodický přístup lepší než ten druhý, ale zaměřují se na různé druhy obtíží, které mohou mít žáci při učení a utváření představ o vybraných algoritmických konceptech při aplikování uvedených metodických přístupů. Při „unplugged“ aktivitách si žáci jednodušeji uvědomovali, co prakticky dělají, a sami se dostali k závěrům, které si žáci, kteří řešili úlohy pouze ve Scratch, neuvědomovali. Jednalo se například o přechod ze zápisu algoritmu pomocí šipek, které představovali základní příkazy, k zápisu algoritmu pomocí opakování. Téměř všichni žáci sami zjistili, že psát instrukce pomocí šipek stále dokola zabírá moc času, a chtěli si automaticky ušetřit práci tím, že vytvoří kratší a efektivnější zápis algoritmu. Žáci si při „unplugged“ aktivitách také lépe utvářeli představy o smyslu a funkčním principu příkazů s testovacími podmínkami. K tomuto jevu u žáků, kteří řešili pouze úlohy ve Scratch, nedocházelo, jelikož mnozí z nich nedokázali pochopit, proč by se měli učit používat ve virtuálním blokově orientovaném programovacím prostředí jiné složitější příkazy, když lze dané úlohy vyřešit pouze pomocí sekvence základních příkazů, kdy algoritmus jednoduše sestavují pomocí přesouvání bloků

[14] GAIO, A. Programming for 3rd graders, Scratch-based or Unplugged? University of Palermo, Department of Mathematics and Computer Science. [online]. Dostupné z: https://keynote.conference-services.net/resources/444/5118/pdf

příkazů. Naopak výzkum ukázal na to, že žáci, kteří řešili úlohy pouze ve Scratch, dokázali rychleji najít a opravit chyby v sestaveném algoritmu. Při „unplugged“ aktivitách se zase žáci nedokázali oprostit od vizuálního vnímání zadané úlohy a víceméně nevědomě se dostali ke správnému řešení úlohy mnohdy i se špatnými instrukcemi. Ve virtuálním programovacím prostředí žáci při sledování průběhu programu nacházeli chybu rychleji, jelikož při simulaci sestaveného algoritmu viděli, kdy a jak počítač špatně zareagoval. Výsledky výzkumu ukazují, že upřednostňování jednoho ze zmiňovaných metodických přístupů před druhým by nejspíš nebylo správné a bylo by spíše efektivnější tyto metodické přístupy vhodně kombinovat.

Rakousko

Velmi často se výuka programování zaměřuje na použití specifických algoritmům v konkrétních případech, které se považují za důležité v edukaci nebo praxi. Na rozdíl od tohoto přístupu ukazuje Gerald Futschek, profesor Institutu softwarových technologií a interaktivních systémů na Vienna University of Technology, způsob učení funkčním principům a konceptům algoritmů, kterým žáci mnohem lépe porozumí, a navíc ještě zábavnou formou.[15] Jeho myšlenkou je zapojit co nejvíce žáků do hraní si na algoritmy, které oni sami navrhnou. Úkolem učitele je žákům představit daný problém a následně jim pokládat správné zvolené otázky, které udrží jejich myšlenkový pochod s cílem vytvořit funkční algoritmy, které tyto problémy vyřeší. Učitel také motivuje žáky, aby zlepšili své algoritmy s cílem najít efektivnější řešení. Příkladem takové úlohy může být například nalezení maximální hodnoty (největšího čísla), kdy každý student v posluchárně představuje nějaké číslo nebo hraní si na roboty, kdy jeden žák na sebe bere roli robota a druhý roli navigátora. Takto se žáci nevědomě během toho, co si víceméně hrají, učí základy algoritmického myšlení, společně navrhují řešení a následně prožívají navržené algoritmy tím, že si je přehrávají a následně je opravují s cílem sestavit efektivnější a obecně funkční algoritmus. Pro začátečníky není důležitá znalost specifických algoritmů, důležitá je schopnost rozumět funkčním principům algoritmů a najít nebo vytvořit vlastní algoritmy pro nové problémy. Začátečníci by nejprve měli rozumět tomu, že algoritmus přesně určuje, co

[15] FUTSCHEK, G., MOSCHITZ, J. Developing Algorithmic Thinking by Inventing and Playing Algorithms. Constructionism 2010. [online] 2010. Dostupné z: https://publik.tuwien.ac.at/files/PubDat_187461.pdf

dělat v každé situaci. Tohle je možné zažít právě ve hře, která přesně následuje skript vlastnoručně vytvořeného algoritmu. Hraní na algoritmy je tudíž vhodnou metodou k naučení se funkčních principů jednotlivých algoritmických konceptů sice pomalejší, zato však efektivnější cestou.

4.1.2 Analýza dostupných úloh a didaktických materiálů

Součástí diplomové práce je i analýza dostupných úloh a didaktických materiálů, které mohou přispět k rozvíjení algoritmických dovedností žáků 1. stupně ZŠ a které se týkají navrhování a sestavování algoritmů s použitím testovacích podmínek.

Bobřík informatiky

Bobřík informatiky je informatická soutěž pro žáky 4. až 9. ročníků základních škol a studenty středních škol, která je podporovaná MŠMT a Jednotou školských informatiků. Soutěž má za cíl rozvíjet informatické myšlení žáků a ukázat, že informatické problémy se objevují v každodenním životě. Informatické myšlení pak přispívá k rozvoji schopnosti abstrahovat, analyzovat, hledat vhodné strategie pro řešení problémů a ověřovat je v praxi.[16] Archiv testů lze použít i při školní výuce informatiky.[17]

Soutěžní otázky se liší svojí náročností, zaměřením, způsobem řešení a tematicky je možné je rozdělit do následujících oblastí:[18]

- **algoritmizace** a programování
- **porozumění informacím** a jejich reprezentacím (kódování, jazyk, šifrování), strukturám (grafy, mapy), procesům
- **řešení problémů** (hledání strategií, logika, matematické základy informatiky)
- **digitální gramotnost**, informační technologie v každodenním životě, technické otázky, společenské souvislosti používání technologií

[16] WAGNER, J. Bobřík informatiky – výběr úloh z národních kol soutěže 2010 až 2014. Pedagogické.info [online] 14. 3. 2017. Dostupné z: http://www.pedagogicke.info/2017/03/bobrik-informatiky-vyber-uloh-z.html
[17] BOBŘÍK INFORMATIKY – Archiv testů. [online]. Dostupné z: https://www.ibobr.cz/test/archiv
[18] BOBŘÍK INFORMATIKY - Informace o soutěži – menu. [online]. Dostupné z: https://www.ibobr.cz/o-soutezi

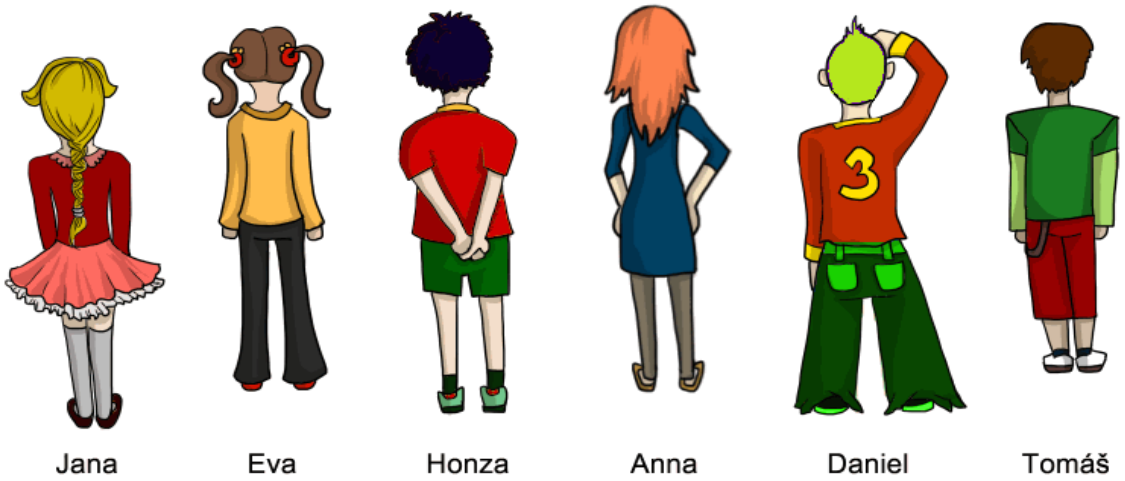
Žáci jsou v soutěži rozdělení podle věku do jednotlivých kategorií:

- **Mini -** pro 4. - 5. ročník ZŠ
- **Benjamin -** pro 6. - 7. ročník ZŠ (prima, sekunda)
- **Kadet -** pro 8. - 9. ročník ZŠ (tercie, kvarta)
- **Junior -** pro 1. - 2. ročník SŠ (kvinta, sexta)
- **Senior -** pro 3. - 4. ročník SŠ (septima, oktáva)

Analyzované úlohy v Bobříku informatiky, které se týkaly testovacích podmínek, se víceméně nezaměřují na rozvoj schopnosti použít příkazy s testovacími podmínkami při sestavování algoritmů, ale spíše na to, jak by tyto schopnosti a dovednosti mohly žákům pomoci při řešení tohoto typu úloh. Ukázkou jedné takové úlohy je úloha č. 3 v kategorii Mini v ročníku 2016.

Ukázka úlohy 1: Kdo rozbil okno?

Šest dětí si hrálo na dvorku. Jeden z nich hodil míč a rozbil okno pana Doskočila. Když pan Doskočil přišel k rozbitému oknu, viděl pachatele utíkat pryč ze dvorku. Pamatuje si jen to, že pachatel měl červené tričko nebo mikinu a krátké tmavé vlasy.



Obr. 1 – Bobřík informatiky, úloha č. 3, kategorie Mini, ročník 2016

Vybraná úloha se týká charakteristiky objektu a splnění podmínek. Každé z dětí (Obr. 1) má nějakou barvu vlasů, barvu košile a délku vlasů. Hodnoty těchto údajů však mohou být pro každé různé. Například Jana i Eva mají nějakou barvu vlasů, ale liší se jejich hodnoty, Jana je blond a Eva má vlasy hnědé.

Tabulka č. 2 – Charakteristika dětí z obr. 1

Jméno	**Vrchní díl oblečení**	**Délka vlasů**	**Barva vlasů**
Jana	Červená	Dlouhé	Světlá
Eva	Žlutá	Dlouhé	Tmavá
Honza	Červená	Krátké	Tmavá
Anna	Modrá	Dlouhé	Zrzavá
Daniel	Červená	Krátké	Zelená
Tomáš	Zelená	Krátké	Tmavá

Žáci najdou řešení této úlohy, pokud najdou jedno dítě, které splňuje všechny podmínky třídění v této úloze. Podmínku č. 1 (červené triko nebo mikina) splňují Jana, Honza, a Daniel. Z těchto tří dětí splňuje druhou podmínku (krátké vlasy) Honza a Daniel a třetí podmínku (tmavé vlasy) pouze Honza.

Tato úloha by byla také vhodná jako inspirace pro „unplugged“ aktivitu. Jeden žák by měl za úkol ze skupiny jiných žáků „vytřídit“ vybraného žáka právě pomocí testovacích podmínek. V takovémto případě žáci řeší skutečný problém z reálného života a s řešením nemají větší problémy. Zkušeností autora diplomové práce ukazují, že při řešení úloh v online prostředí Bobříka informatiky mají žáci, zejména mladší žáci, problémy pochopit samotné zadání úlohy a to, co mají vlastně řešit. To nejspíše souvisí s jejich nedostatečnou úrovní čtenářské gramotnosti, jejíž rozvoj pak samozřejmě úzce souvisí s rozvojem informatického myšlení.

Matematický klokan

Dalším zdrojem úloh, které by mohly přispět k rozvíjení algoritmických dovedností žáků 1. stupně ZŠ, jsou úlohy zařazené do Matematického klokana. Matematický klokan je

mezinárodní matematická soutěž vytvořená podle obdobné soutěže, která byla v osmdesátých letech minulého století pořádána v Austrálii.mV ČR se soutěž Matematický klokan konala poprvé v roce 1995 a jejím pořadatelem v ČR je Jednota českých matematiků a fyziků ve spolupráci s Katedrou matematiky PdF UP a Katedrou algebry a geometrie PřF UP v Olomouci.[19]

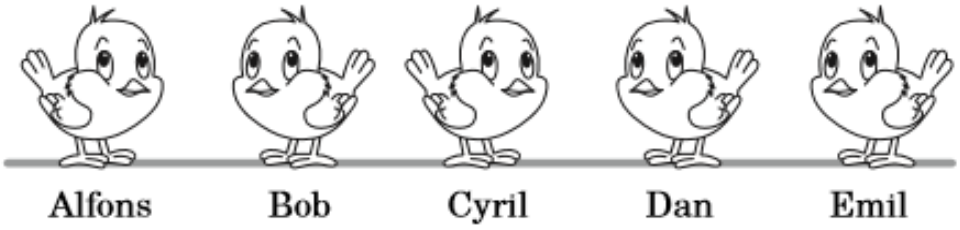
Soutěžící jsou podle věku rozděleni do 6 kategorií:

Cvrček - (2. -3. ročník ZŠ) **Klokánek** (4. - 5. ročník ZŠ),
Benjamín (6. - 7. ročník ZŠ) **Kadet** (8. - 9. ročník ZŠ)
Junior (1. - 2. ročník SŠ) **Student** (3. - 4. ročník SŠ)

Analyzovány byly úlohy z kategorie Cvrček a Klokánek ročníku 2015 a 2016. Počet úloh, které by patřily do oblasti informatiky a mohly by rozvíjet dovednosti spojené s použitím příkazů s testovacími podmínkami při sestavování algoritmů, bylo pochopitelně mnohem méně než v Bobříku informatiky, a víceméně se týkaly třídění informací. Jako ukázka analyzovaných úloh je uvedena následující úloha.

Ukázka úlohy 2 - Matematický klokan 2016, kategorie Klokánek, úloha č. 24

Na větvi sedělo pět vrabců (podívej se na obrázek). Každý zacvrlikal tolikrát, kolik stojí vrabců na té straně, na kterou je natočený. Například vrabec Alfons zacvrlikal čtyřikrát. Jeden z vrabců se otočil na opačnou stranu. Opět každý vrabec zacvrlikal tolikrát, kolik stojí vrabců na straně, na kterou je natočený. Tentokrát vrabci zacvrlikali vícekrát než napoprvé. Který z vrabců se natočil na druhou stranu?



Obr. 2 - Matematický klokan 2016, kategorie Klokánek, úloha č. 24

[19] MATEMATICKÝ KLOKAN – Informace o soutěži [online]. Dostupné z: http://matematickyklokan.net/index.php/o-soutezi/informace-o-soutezi

Při řešení této úlohy je důležité, aby si žáci uvědomili, jaké události nastanou při otočení každého vrabce, což například úzce souvisí s pochopením testovací podmínky „Když“.

Když se otočí Alfons, zacvrliká 0x místo 4x, což je méněkrát.
Když se otočí Bob, zacvrliká 3x místo 1x, což je **vícekrát**.
Když se otočí Cyril, zacvrliká 2x, stejně jako před tím, než se otočil.
Když se otočí Dan, zacvrliká 1x místo 3x, což je méněkrát.
Když se otočí Emil, zacvrliká 0x místo 4x, což je méněkrát.
Pro správné vyřešení nejen této úlohy z Matematického klokana platí to, co pro úlohy v Bobříku informatiky. Je velmi důležité, aby žáci pochopili samotné zadání úlohy a to, co mají vlastně řešit. V opačném případě mohou být výsledky zkreslené.

Ukázka úlohy 3: Tim The Train

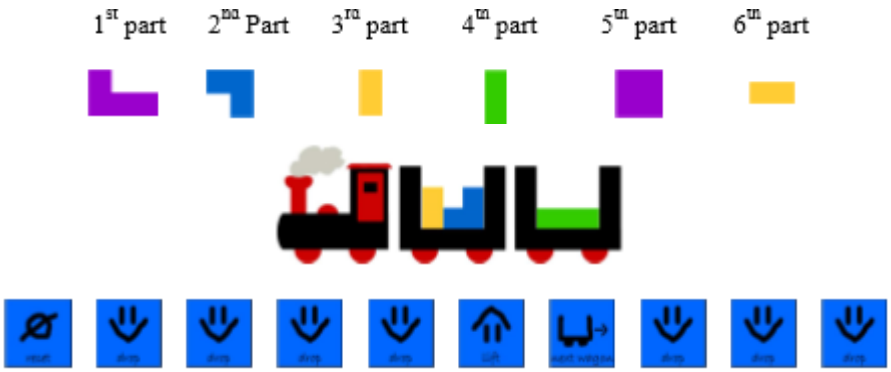
Návrh „unplugged“ aktivit byl z velké části inspirován sadou na sebe navazujících úloh Tim The Train[20], která je určena pro žáky 1. stupně ZŠ a který pomáhá žákům s rozvojem algoritmického myšlení a s přechodem od reálného prostředí s materiálními předměty do virtuálního prostředí umožňujícího sestavit jednoduché programy. Při výuce pomocí scénáře Tim the Train žáci nejprve pracují s materiálními předměty a po té přecházejí z tohoto hravého prostředí do světa počítačového programování v programovacím prostředí Scratch. Pomocí řady úkolů se žáci učí základům algoritmického myšlení a pochopit základní algoritmické koncepty. Smyslem Tim the Train je propojit reálný (materiální) a virtuální svět, a tím pomoci začátečníkům s abstraktním myšlením při programování v programovacím prostředí.

Žáci pro řešení úloh nepotřebují žádné speciální vstupní znalosti, není ani nutné, aby uměli číst, algoritmy sestavují pomocí kartiček se symboly (Obr. 3). Počet symbolů, které představují jednotlivé příkazy a tvoří instrukční sadu, je malý, aby se žáci co nejvíce soustředili pouze na základní algoritmické koncepty.

Mezi základní algoritmické koncepce, se kterými se žáci při plnění úloh scénáře Tim the Train seznamují, patří:

[20] FUTSCHEK, G., MOSCHITZ, J. Learning algorithmic thinking with tangible objects eases transition to computer programming. In International Conference on Informatics in Schools: Situation, Evolution, and Perspectives [online] 2011. Dostupné z: https://publik.tuwien.ac.at/files/PubDat_199953.pdf

- Basic commands – basic actions - Základní příkazy - základní akce
- Sequence of commands – Sekvence (posloupnost) příkazů
- Alternative of commands (if) – Alternativní příkazy (když)
- Iteration of commands (loop) – Opakování příkazů (cykly)



Obr. 3 – Ukázka částí nákladu, vlaku s naloženými vagóny a sady symbolů (příkazů)

Hlavním účelem příkazu v podobě symbolu je, že při jeho spuštění se vyvolá předem definovaná akce. Žák musí pochopit, že příkaz je vlastně abstrakcí akce. Nové příkazové karty v podobě dalších symbolů mohou být také vytvořeny samotnými žáky. Někteří žáci dokáží vytvořit zcela nové akce (funkce) a navrhnout pro tyto akce nové symboly. Při řešení jednotlivých úloh mají žáci za úkol definovat, za jakých podmínek bude sestavený algoritmus funkční, a za kterých ne. Žáci také sami zjišťují, které části algoritmu se opakují, tyto části algoritmu od sebe oddělují a následně intuitivně vytvářejí cykly nebo dokonce vnořené cykly.

Úlohy s Ozobotem

Aktivity s Ozobotem mohou rovněž přispět k rozvoji algoritmického myšlení žáků. Ozobot je malý robot, který se dokáže pohybovat po různých druzích povrchu. Je schopen následovat nakreslené čáry a křivky, reagovat na barevné kódy, měnit rychlost, anebo se díky naprogramování v dostupných aplikacích pro Ozobota pohybovat volně. Ozobot při svém pohybu hlídá linii, kterou sleduje svými pěti senzory, a jeho LED dioda mění barvu podle barvy dráhy. Prostřední senzor je senzor pro barvy. Umí rozeznat červenou, modrou a zelenou barvu. A vzhledem k tomu, že se všechny barvy dají složit z těchto tří barev, tak

je Ozobot umí také rozpoznat. Ozobot reaguje na změnu barvy čáry, po které se pohybuje, a díky tomu můžeme jeho chování pomocí vizuálních barevných kódů (Ozokódů) jednoduše naprogramovat. Ozobot je vybaven „vlastní inteligencí“ založenou na náhodně generovaných rozhodnutích. Pokud narazí na křižovatku cesty v podobě barevných čar, náhodně se rozhodne, po které cestě se vydá.

Ozobot se dá také programovat pomocí počítače, chytrého telefonu či tabletu. V tom případě je nutné si do smartphonu nebo tabletu bezplatně stáhnout a nainstalovat aplikaci Ozogroove (iOS, Android), nebo Ozobota naprogramovat online v bezplatném blokově orientovaném programovacím prostředí OzoBlockly, kde se žáci učí správným programovacím návykům.

V dnešní době jsou takovéto stroje využívané v různých průmyslových odvětvích například pro přepravu materiálů ve skladech, továrních halách, ale také třeba pro dopravu jídla v restauracích nebo léků v nemocnicích. Některé z nejdříve používaných automatizovaně vedených vozidel (AGV) se pohybovaly právě podle čar nakreslených na podlahu nebo strop. První stroj byl zaveden v roce 1950 a v té době to bylo odtahovací vozidlo, které následovalo dráty na podlaze.[21]

Na webových stránkách internetového prodejce Easystore[22] lze najít velmi podrobný popis Ozobota, rychlé návody a tipy, jak s Ozobotem pracovat, výukové lekce a aktivity přeložené do českého jazyka, odkazy na online aplikace určené pro práci s Ozobotem a také odkazy na recenze.[23]

Analýza úloh s Ozobotem byla zaměřena na vyhledání vhodných úloh pro návrh úvodní motivační hodiny pedagogického experimentu. Analyzovány byly úlohy z uživatelského portálu Ozobot - Lesson Library.[24] Navržené aktivity v rámci úvodní motivační hodiny jsou uvedeny v kapitole 4.2 - Návrh úvodních motivačních aktivit s Ozobotem.

[21] OZOBOT – Basic Training Lesson 1 - https://portal.ozobot.com/lessons/detail/basic-training-1
[22] https://www.easystore.cz/ozobot-bit-inteligentni-minibot-bily.html#tipy
[23] ČERNÝ, M. Ozobot – malý, ale šikovný. [online] 2014. Dostupné z: http://robodoupe.cz/2014/ozobot-maly-ale-sikovny/
[24] OZOBOT - Lesson Library [online]. Dostupné z: https://portal.ozobot.com/lessons

Úlohy v Code.org

Code.org je bezplatná on-line aplikace, která byla spuštěna roku 2013 a je zaměřena na výuku programování pro různé věkové kategorie od mateřských škol až po střední školy. Pomocí různých kurzů, ve kterých se řeší jednotlivé úlohy, si žáci osvojují základní programovací návyky a učí se základům programování. Podrobná analýza velkého množství úloh v Code.org sloužila zejména pro výběr aktivit ve vybraném virtuálním blokově orientovaném programovacím prostředí, které bezprostředně navazují na navržené „unplugged" aktivity. Vybrané úlohy v Code.org navazující na navržené „unplugged" aktivity jsou uvedeny v kapitole 4.4.2 - Návrh úloh a aktivit v Code.org.

On-line hry pre deti

Analyzovány byly také online úlohy doc. PaedDr. Moniky Tomcsányiové z Katedry základov a vyučovania informatiky Fakulty matematiky, fyziky a informatiky UK v Bratislavě.[25] Výsledky analýzy těchto úloh však v diplomové práci použity nebyly.

4.2 Návrh úvodních motivačních aktivit s Ozobotem

Do navrženého pedagogického experimentu budou, jako úvodní a motivační část navržených aktivit, zařazeny aktivity s Ozobotem ze Základního tréninku – lekce 1. Cílem těchto aktivit bude žáky na úvod motivovat, ale také vést k tomu, aby si uvědomili a uměli vysvětlit, jak se Ozobot chová, jak je naprogramován a jakým způsobem reaguje na sekvenci barev (Ozokódy). Žáci se také seznámí s Ozokódy, se speciálním programovacím jazykem, kterému Ozobot rozumí, a pomocí kterých lze Ozobota ovládat, a které následně použijí k naprogramování Ozobota. Žáci se také teoreticky seznámí s praktickým využitím automatizovaně vedených vozidel (AGV).

Pro řešení úloh žáci použijí následující pomůcky:

Ozobot, čistý papír A4, barevné fixy (červený, modrý, zelený, černý), tiskopisy ze Základního tréninku – lekce 1: přehled barevných Ozokódů (viz Příloha 1), dráha s prázdnými místy pro Ozokódy (viz Příloha 2), dráha s Ozokódy (viz Příloha 3), dráha pro úlohu „Cesta k obchodu" (viz Příloha 4).

[25] TOMCSÁNYIOVÁ, M. On-line hry pre deti [online]. Dostupné z: http://edi.fmph.uniba.sk/~tomcsanyiova/

Při plnění jednotlivých úloh budou žáci ústně v rámci řízeného rozhovoru hromadně odpovídat na připravené otázky a písemně doplňovat nedokončené věty (viz Příloha 5):
Když se Ozobot pohybuje po červené dráze, rozsvítí se…

Když se Ozobot pohybuje po modré dráze, rozsvítí se…

Když se Ozobot pohybuje po zelené dráze, rozsvítí se…

Když Ozobot narazí na pořadí (sekvenci) barev (modrá, černá, modrá), tak…

Když Ozobot narazí na pořadí (sekvenci) barev (červená, černá, červená), tak…

Když Ozobot narazí na pořadí (sekvenci) barev (modrá, červená, modrá), tak…

Když dráha dál nepokračuje (končí), tak Ozobot…

Když přijede Ozobot na křižovatku, jakým směrem se vydá?

Jak Ozobot rozpozná barvy?

Přemýšlí robot?

4.3 Návrh aktivit a úloh bez počítače („unplugged“ aktivit)

Téma algoritmizace je obecně mezi rodičovskou veřejností považováno za složité, nezábavné a neatraktivní. Výuka programování pak často bývá zaměřena na používání specifických algoritmů a algoritmických konceptů ve vhodném programovacím prostředí v podobě počítačového softwaru. Samotné algoritmické myšlení je ovlivněno řadou kognitivních faktorů, jako je abstraktní a logické myšlení, myšlení ve strukturách, kreativita nebo kompetence pro řešení problémů. Vzhledem k těmto faktorům není jednoduché se algoritmickému myšlení naučit, a proto je nutné přistupovat k výuce základů programování citlivě se správně zvolenými metodickými postupy zvláště pak u mladších žáků nebo začátečníků.

V této kapitole se diplomová práce zaměřuje na návrh scénáře na sebe navazujících úloh pro výuku vybraných základních algoritmických konceptů bez použití počítače. V navržených „unplugged“ aktivitách si žáci mimo jiné „hrají na algoritmy“ nebo manuálně přehrávají sestavené algoritmy. Tímto způsobem by žáci měli lépe porozumět funkčním principům vybraných algoritmických konceptů a naučit se přemýšlet jako stroj, který lze

naprogramovat velmi jednoduchou sadou instrukcí. Pro pochopení vybraných algoritmických konceptů je důležité klást důraz na to, aby princip a smysl použití jednotlivých příkazů dokázali žáci pomocí správně zvolených otázek a na sebe navazujících úkolů objevit sami.[26]

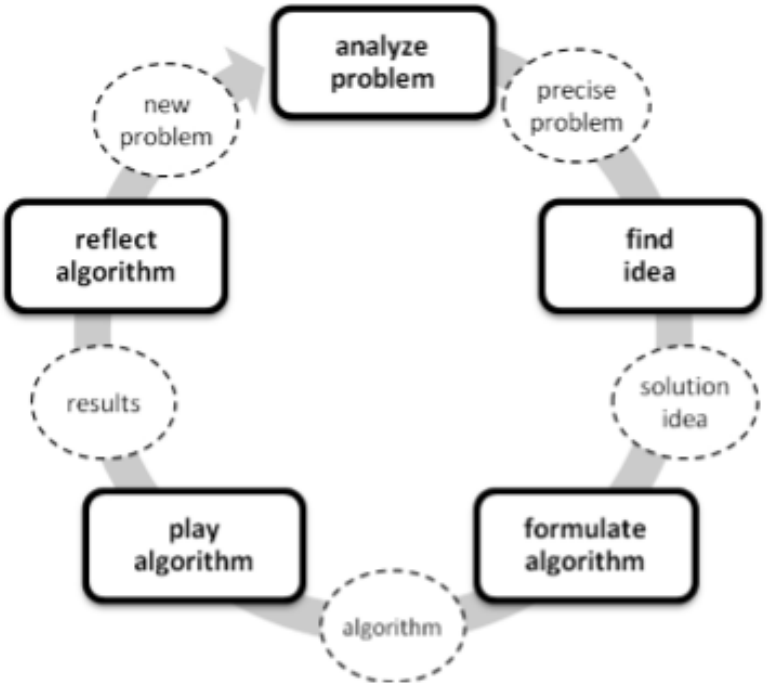
Navržené „unplugged" aktivity se zaměřují na samotné pochopení funkčních principů vybraných algoritmických konceptů zábavnou formou, která by žáky neměla od témat tykajících se algoritmizace odradit. Hlavní myšlenkou navržených „unplugged" aktivit je zapojení co nejvíce žáků do „hraní si na algoritmy" v podobě manuálního přehrávání sestavených algoritmů, které sami navrhnou. Úkolem učitele je studentům problém srozumitelně představit a pomocí vhodně zvolených otázek a úkolů navozovat další obtížnější problémy, které žáci musí postupně řešit pomocí algoritmů se složitějšími příkazy. Takovýto postup motivuje žáky udržovat pozornost, sestavovat funkční algoritmy, které zadané problémy vyřeší a zlepšovat své algoritmické dovednosti.

Pro jednodušší a rychlejší nalezení řešení jednotlivých úloh je vhodné, aby žáci pracovali ve skupinách, kdy o daném dílčím problému mohou diskutovat a obhajovat svoje názory. V tomto případě dochází i k vrstevnickému učení, kdy se slabší žáci učí jak řešit problémy od svých spolužáků. Navržené „unplugged" aktivity pomáhají pochopit vybrané algoritmické koncepty i těm žákům, kteří mají problémy s algoritmickým myšlením nebo těm, kteří preferují vizuální nebo kinestetický učební styl. Tyto typy žáků si lépe osvojují učební látku, pokud se mohou něčeho dotýkat a pracovat s materiálními předměty v reálném světě, a proto je pro ně rozvoj základních algoritmických dovedností, a tím také rozvoj algoritmického myšlení, příjemnější bez abstraktního programovacího prostředí.

Jednotlivé „unplugged" aktivity byly navržené podle modelu, který rozděluje edukační proces výuky vybraných algoritmických konceptů na 5 hlavních kroků (Obr. 4).[27]

[26] FUTSCHEK, G., MOSCHITZ, J. Learning algorithmic thinking with tangible objects eases transition to computer programming. In International Conference on Informatics in Schools: Situation, Evolution, and Perspectives [online] 2011. Dostupné z: https://publik.tuwien.ac.at/files/PubDat_199953.pdf

[27] FUTSCHEK, G., MOSCHITZ, J. Developing Algorithmic Thinking by Inventing and Playing Algorithms. Constructionism 2010. [online] 2010. Dostupné z: https://publik.tuwien.ac.at/files/PubDat_187461.pdf



Obr. 4 - Proces učení sestavování algoritmů (G. Futschek & J. Moschitz, 2010, s. 4)

Krok 1 - Analyzovat problém - upřesnění problému

Prvním krokem je zjistit více detailů o problému. V této části žáci formulují podstatu úlohy, zkouší najít hlavní problém a eventuálně jej rozdělit na menší dílčí problémy.

Krok – 2 - Nalézt myšlenku - nápad na řešení

V tomto kroku je kladen důraz na kreativitu, kdy žáci přemýšlí o tom, jak daný problém vyřešit. Žáci se ve skupinách snaží jednotlivé nápady na řešení zhodnotit a domluvit se na tom, který nápad může daný problém vyřešit nejlépe, nebo který nápad bude nejjednodušší zrealizovat.

Krok 3 - Formulovat algoritmus - algoritmus

V tomto kroku je cílem přesně formulovat vybraný návrh algoritmu. Formulace algoritmu probíhá prostřednictvím diskuse žáků s ostatními skupinami a vyučujícím. Začátečníci mohou řešení popsat slovně, pokročilejší žáci mohou přesnou formulaci napsat.

Krok 4 – Hraní si na algoritmy - výsledky

Hlavním cílem této části je zjistit, jestli vůbec algoritmus funguje a jak dobře funguje. Žáci se v této fázi učí, že ne všechny nápady fungují dokonale, a snaží se objevit chyby v sestaveném algoritmu. Při hraní si na algoritmy si také žáci jednodušeji uvědomují, které akce se provedou v jednotlivých krocích algoritmu.

Krok 5 - Reflexe algoritmu - nový problém

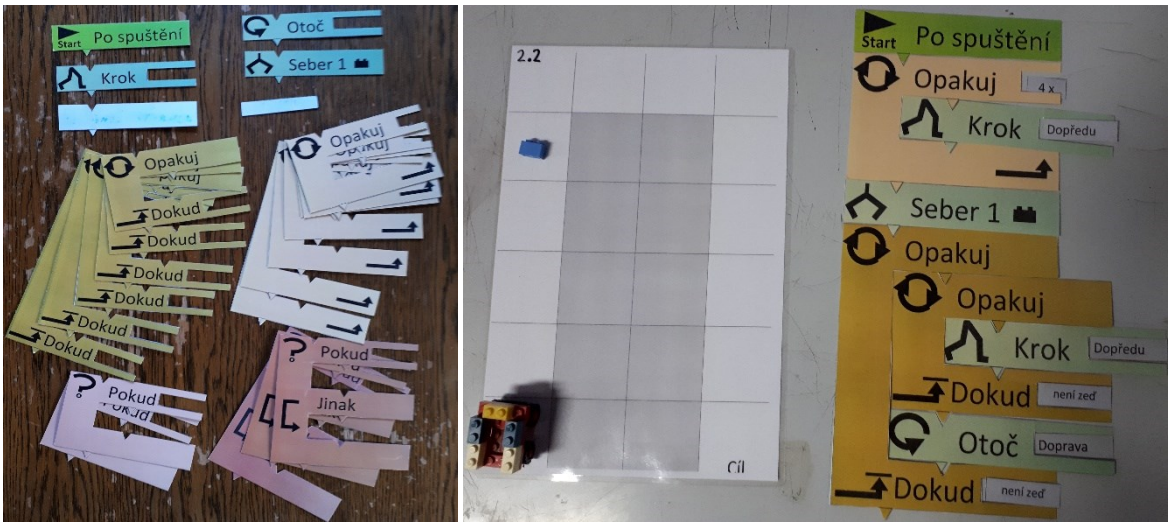
Cílem reflexe algoritmu je zjistit efektivitu algoritmu, to znamená, jestli je algoritmus například rychlý, nebo pomalý, nebo jestli nedělá zbytečné kroky. Na základě těchto zjištění pak žáci navrhují jeho vylepšení. Žáci například sami objevují, že ty části algoritmů, které se opakují, je výhodnější a rychlejší zapsat pomocí cyklu. Výsledkem reflexe jsou pak nové problémy k řešení, takže proces začíná znovu od začátku.

Sada úloh - Automatický sběrač odpadků

V této úloze žáci nejprve pracují s materiálními předměty a po té přecházejí z hravého materiálního prostředí do světa počítačového programování. Pomocí řady úkolů bez použití počítače se žáci učí základům algoritmického myšlení a pochopit smysl a funkční principy vybraných algoritmických konceptů. Cílem navržených „unplugged“ aktivit je vytvořit propojení mezi reálným světem a virtuálním světem počítačového programování, a tím pomoci začátečníkům s abstraktním myšlením při programování v programovacím prostředí. Materiální učební pomůcky tvoří malé vozidlo z Lega (Obr. 5), které si eventuálně mohou žáci sestavit sami a které představuje sběrač odpadků, dále pak připravené dráhy, po kterých se má vozidlo pohybovat, malé kostičky Lega představující odpadky a kartičky s příkazy, které tvoří sadu instrukcí pro ovládání vozidla.

Obr. 5 – Příklad vozidel postavených z Lega

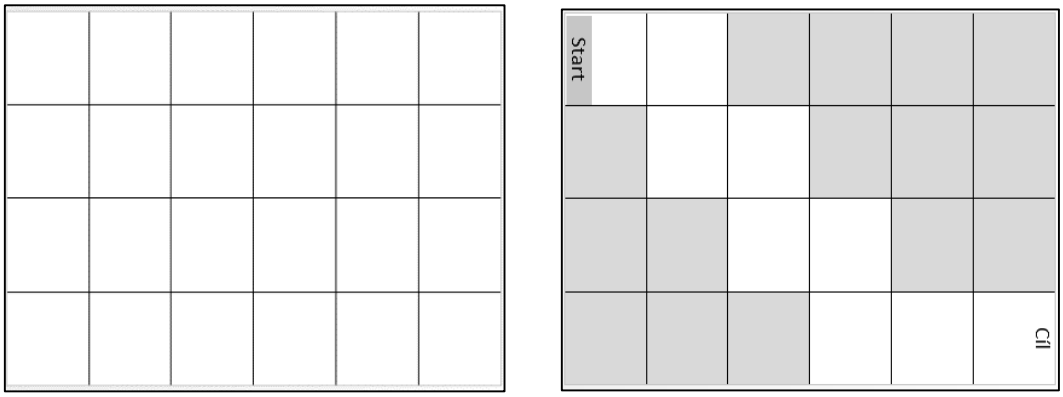
Žáci mají v jednotlivých úlohách za úkol sestavit algoritmus pro pohyb vozidla a sběr odpadků pomocí materiálních kartiček, které představují jednotlivé příkazy (dále jen kartičky s příkazy). Kartičky podle typu příkazu, který představují, mají různý tvar, barvu a obsahují různé symboly a texty. Tvary kartiček jsou navrženy tak, aby se při sestavování algoritmu nemohly spojit dílky, které k sobě ze syntaktického hlediska nepatří. Proto mají jednotlivé kartičky na sobě umístěné výřezy a výčnělky, aby do sebe zapadaly a aby tvořily jakousi skládačku ve smyslu puzzle. Symboly a texty na těchto kartičkách byly zvoleny tak, aby jejich použití při sestavování algoritmu bylo intuitivní a aby co nejvíce připomínaly příkaz, který představují (Obr. 6). Samotný vzhled kartiček je potom podobný vzhledu příkazů blokově orientovaných programovacích jazyků určených pro výuku programování malých dětí (např. Scratch, Code.org). Kartiček by nemělo být mnoho, aby se žáci mohli co nejvíce soustředit pouze na samotné osvojování vybraných algoritmických konceptů.



Obr. 6 – Ukázka jedné sady kartiček s příkazy a z ní sestaveného algoritmu

Kartičky usnadňují žákům převedení konkrétních pojmů v podobě kartiček s příkazy na pojmy abstraktní v podobě akcí, které nastanou po spuštění daného příkazu, a také pomáhají žákům s přechodem z reálného materiálního světa do virtuálního prostředí blokově orientovaného programovacího jazyka. Účelem příkazu v podobě kartiček s příkazy je, že při jeho spuštění se vyvolá předem definovaná akce, takže by žáci měli snáze pochopit, že symbol je vlastně abstrakcí akce.

Velmi důležitou aktivitou pro rozvoj algoritmického myšlení a pro pochopení funkčních principů vybraných algoritmických konceptů je pak manuální přehrávání sestaveného algoritmu samotnými žáky pomocí pomůcek v podobě vozidla, jednotlivých drah a malých kostiček Lega představujících odpad. Jeden žák ze skupiny simuluje pohyb vozidla po dráze a nakládání odpadu tím, že plní postupně jednotlivé příkazy sestaveného algoritmu, které mu čte jiný žák skupiny. Při této aktivitě je důležité, aby žák, který simuluje pohyb vozidla, neviděl sestavený algoritmus a pouze plnil příkazy, které mu zadává jiný žák. Při manuálním přehrávání algoritmu se žáci učí přemýšlet v řeči stroje, uvědomují si, které události nastanou v jednotlivých krocích programu a snaží se objevit eventuální chyby ve svém programu. Jednotlivé dráhy pro vozidlo jsou vytvářené pomocí čtvercové sítě (Obr. 7). Stínováním různých polí čtvercové sítě a eventuálním skládáním čtvercových sítí k sobě se mohou vytvářet různé modifikace tvaru nebo délky dráhy.



Obr. 7 – Ukázka způsobu tvorby jednotlivých drah

Pro rozvoj algoritmického myšlení je pochopení základních algoritmických konceptů nezbytné. Mezi základní algoritmické koncepce, se kterými se žáci při plnění úloh seznamují, patří:

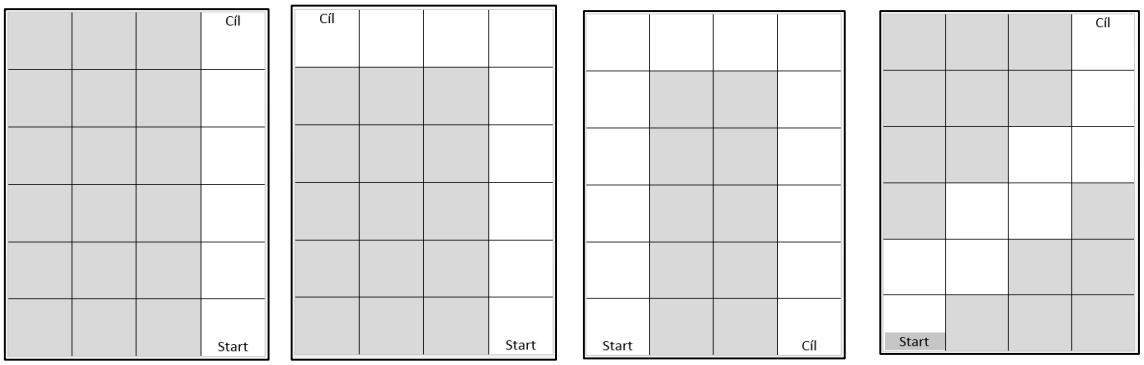
- Základní příkazy - základní akce (Krok, Otoč, Seber)
- Sekvence (posloupnost) příkazů
- Testovací podmínky (Opakuj, Opakuj – Dokud, Když - Potom, Když - Potom - Jinak)
- Abstrakce (zobecnění) příkazů - základní příkazy, testovací podmínky
- Ladění

Další důležité algoritmické koncepty jako například rekurze, parametry, proměnné nebo datové typy jsou úmyslně vynechány, jelikož tyto koncepty vyžadují hlubší abstraktní myšlení, a jsou tedy vhodné spíše pro starší nebo pokročilejší žáky.

Úloha č. 1 – Posloupnost základních příkazů

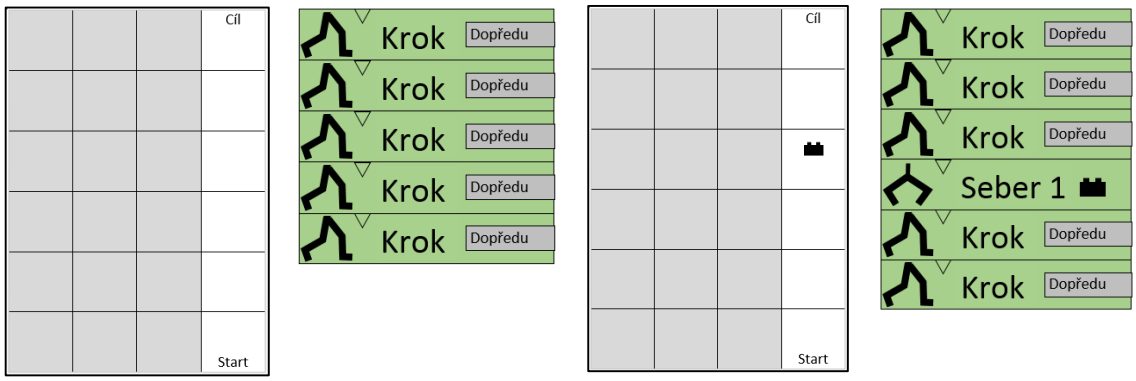
Cílem této úlohy je naučit žáky orientovat se v materiálních kartičkách, které představují základní příkazy (Krok, Otoč, Seber), a pomocí těchto kartiček s příkazy sestavit posloupnost příkazů tak, aby byl sestavený algoritmus funkční pro danou úlohu.

Žáci pracují ve skupinách (nejlépe 3 – 4 žáci) a každá skupina má k dispozici dráhy a základní příkazy (Krok, Otoč, Seber). Na začátku úlohy učitel žákům manuálně předvede, jakým způsobem by se mělo vozidlo pohybovat od startu do cíle a jakým způsobem sbírá kostičky. Při ukázce je důležité, aby žáci pochopili, že se vozidlo po použití kartičky „Krok“ poposune pouze o jedno pole, po použití kartičky „Otoč“ (Doprava, Doleva) se otáčí na místě a po použití kartičky „Seber“ vozidlo sbírá odpad na políčku, na kterém právě stojí. Žáci mají za úkol seřadit kartičky s příkazy tak, aby sestavili algoritmus daného pohybu vozidla. Řazení kartiček s příkazy pod sebe simuluje řazení příkazů ve virtuálním programovacím prostředí.



Obr. 8 – Ukázka možných drah, po kterých se bude pohybovat vozidlo

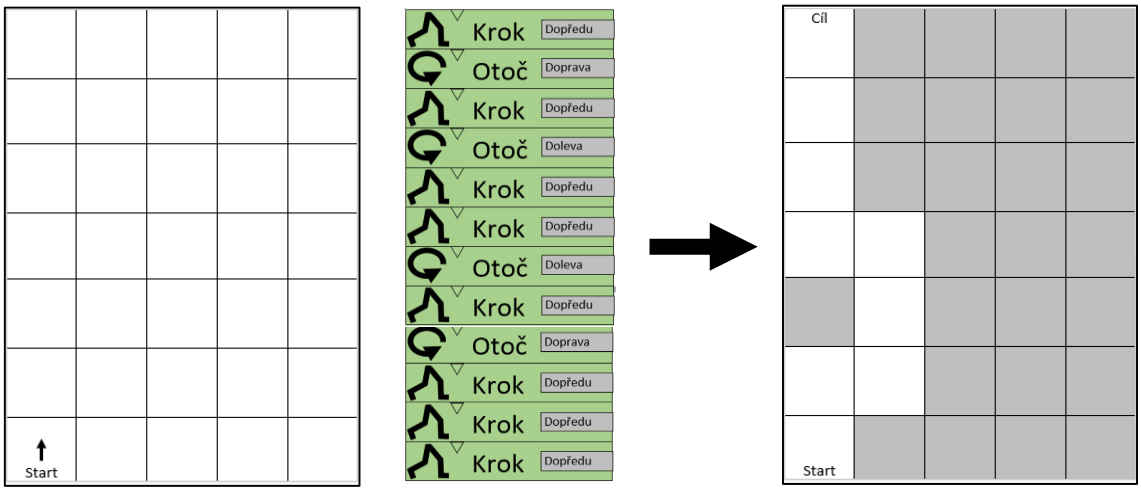
Následně jednotlivým skupinám žáků učitel rozmístí na různá políčka jednotlivých drah kostičky Lega představující odpad a žáci mají za úkol přepracovat sestavené algoritmy tak, aby vozidlo kostičky sebralo (Obr. 9). Je vhodné, aby na začátku úlohy učitel umístil pouze jednu kostičku Lega na jedno vybrané políčko dráhy. Výsledné algoritmy si žáci mezi sebou manuálně přehrávají, snaží se odhalit chyby a odladit chybný algoritmus.



Obr. 9 – Ukázka vybraných drah a k nim řešení algoritmu pohybu vozidla

Úloha č. 2 - Čtení algoritmu

Tato úloha je zaměřená na ověření pochopení posloupnosti základních příkazů. V této úloze mají žáci k dispozici sestavený algoritmus a na prázdné šachovnici, kde je vyznačený pouze „Start“ a směr vozidla, mají za úkol vyznačit dráhu, po které se vozidlo pohybuje a vyznačit cíl, kde po přehrání algoritmu vozidlo zůstane (Obr. 10).



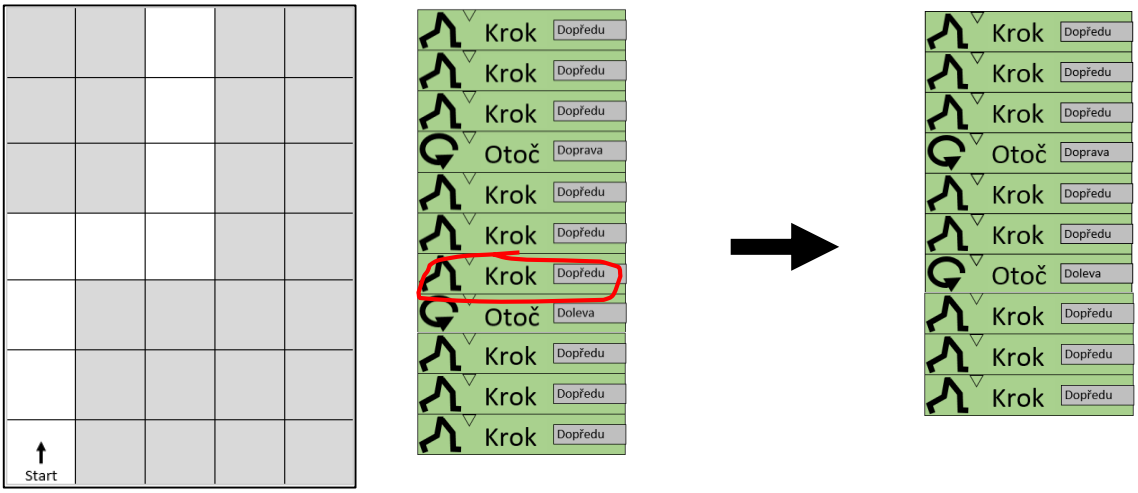
Obr. 10 – Ukázka možného zadání a řešení úlohy č. 2

Žáci mohou po splnění úlohy sestavovat jiné různé algoritmy a zadávat je k řešení jiným skupinám. Výsledky si skupiny mezi sebou porovnávají a pokouší se odhalit a opravit eventuální chybu.

Úloha č. 3 – Ladění algoritmu

Tato úloha je zaměřená na ověření pochopení posloupnosti základních příkazů a na ladění nefunkčního algoritmu. V této úloze dostanou jednotlivé skupiny již vyznačenou dráhu

a k ní chybně zapsaný (nefunkční) algoritmus (Obr. 11). Žáci mají za úkol objevit v dané posloupnosti příkazů chybu a tuto chybu následně opravit (odladit). Výsledky si skupiny mezi sebou porovnávají a pokouší se odhalit a opravit eventuální chybu.



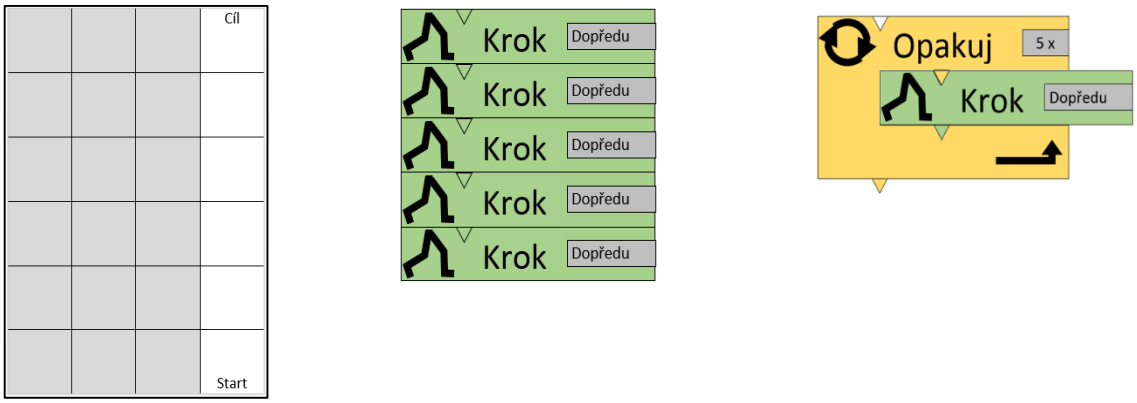
Obr. 11 - Ukázka možného zadání a řešení úlohy č. 3

Jednotlivé skupiny žáků mohou po splnění úlohy vytvářet pro vozidlo jiné dráhy a k nim chybné algoritmy a následně nechat jinou skupinu žáků tyto chybné algoritmy odladit. Takovéto doplňkové a rozšiřující aktivity jsou vhodné zejména pro rychlejší skupiny, které čekají na to, až pomalejší skupiny žáků dořeší předchozí úlohy.

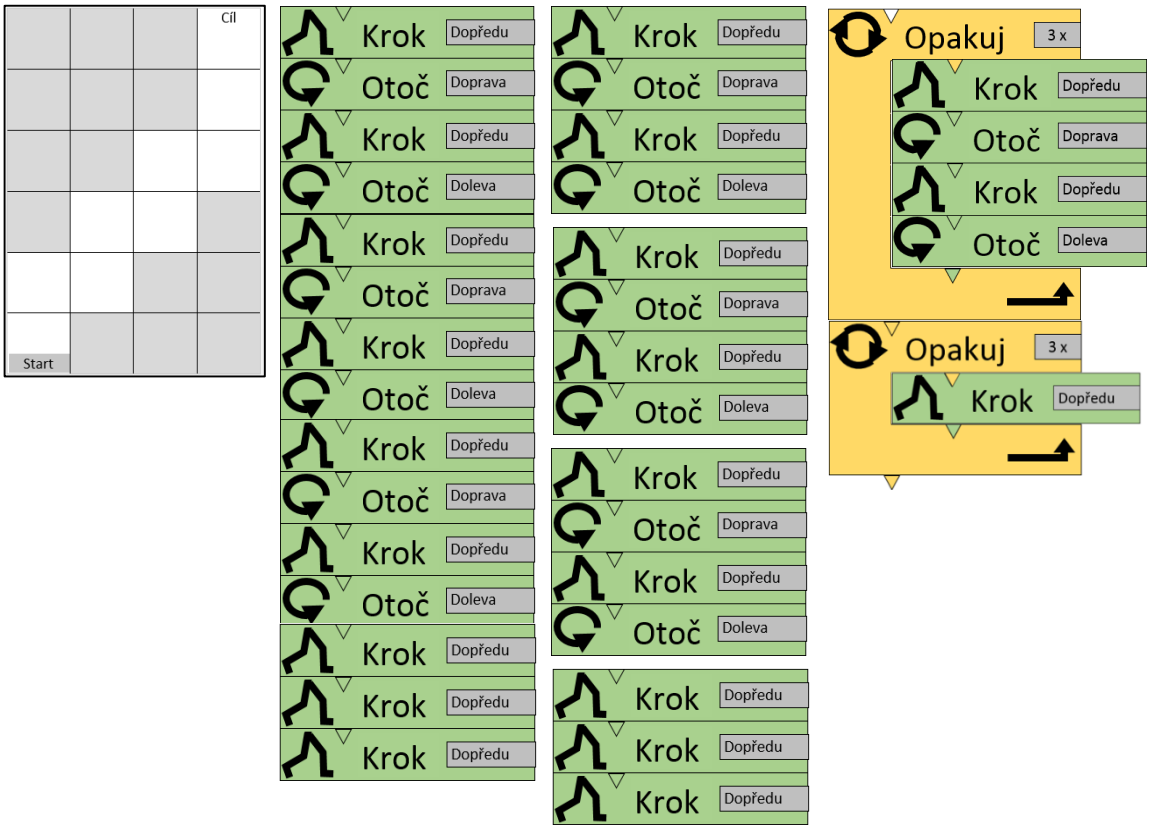
Úloha č. 4 – Algoritmy s cyklem *Opakuj Xkrát*

Tato úloha je zaměřená na pochopení smyslu použití a funkčního principu cyklů (opakování) při sestavování programů. Žáci by měli sami zjistit, že sestavovat algoritmus pomocí stejných příkazů stále dokola zabírá moc času a že pomocí opakovacího příkazu dokáží daný algoritmus zapsat rychleji a s použitím menšího počtu příkazů (Obr. 12).

V této úloze se žáci seznamují s příkazem „*Opakuj Xkrát*“. Snaží se v již sestavených algoritmech z úlohy č. 1 sami objevit příkazy nebo části algoritmu, které se opakují, a následně je přepsat pomocí opakování příkazu nebo části algoritmu. Žáci by měli pochopit, že v takovém případě počítač nebo jiný stroj musí vědět, kolikrát má daný příkaz opakovat, a že použití cyklů dokáže programátorovi ušetřit čas při sestavování počítačového programu.



Obr. 12 – Ukázka zadání a řešení opakování jednoho příkazu



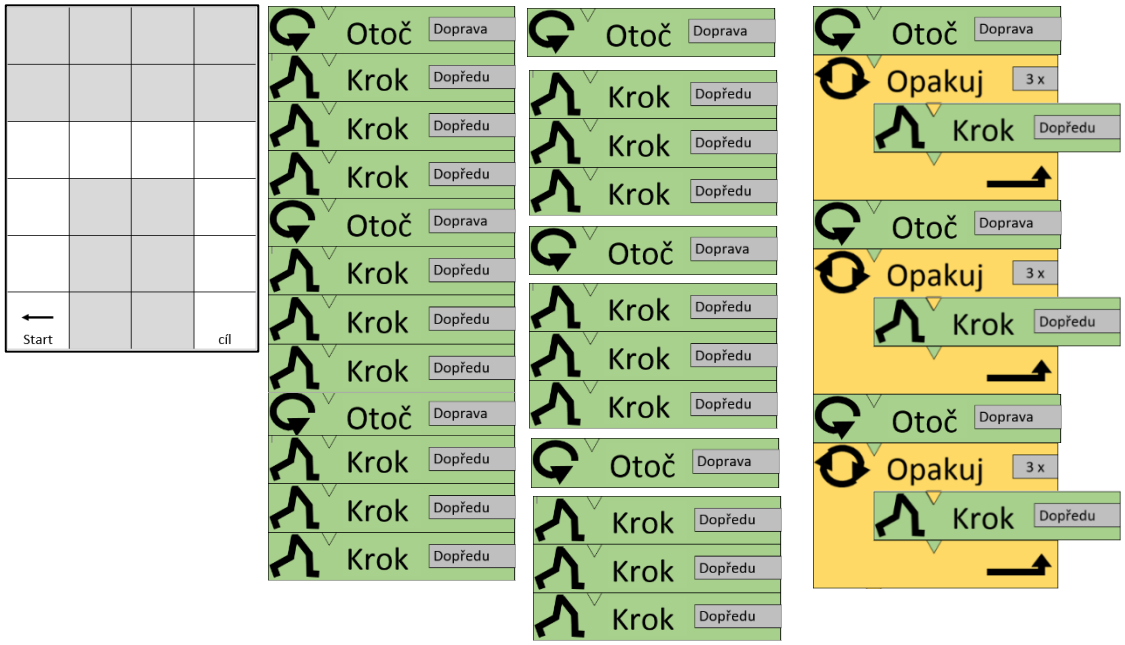
Obr. 13 – Ukázka zadání a postupu při řešení vybrané úlohy s cykly

Předchozí úkol byl víceméně jednoduchý, jelikož se v algoritmu opakoval pouze jeden příkaz. Dalším úkolem je použití příkazu „*Opakuj Xkrát*“ v algoritmu, kde se opakuje skupina příkazů (Obr. 13). Jelikož pochopení cyklů a zejména vnořených cyklů není zejména pro mladší žáky úplně jednoduché, je nezbytně nutné, aby žáci dokázali nejprve najít části algoritmu, které se opakují, a aby také dokázali určit, kolikrát se tato část algoritmu opakuje.

Žáci mají nejprve za úkol příkazy nebo části algoritmu, které se opakují, od sebe oddělit a následně použít pro sestavení funkčního algoritmu kartičku s opakovacím příkazem. Žáci mají po té za úkol porovnat počet kartiček s příkazy, které použili před a po použití opakování.

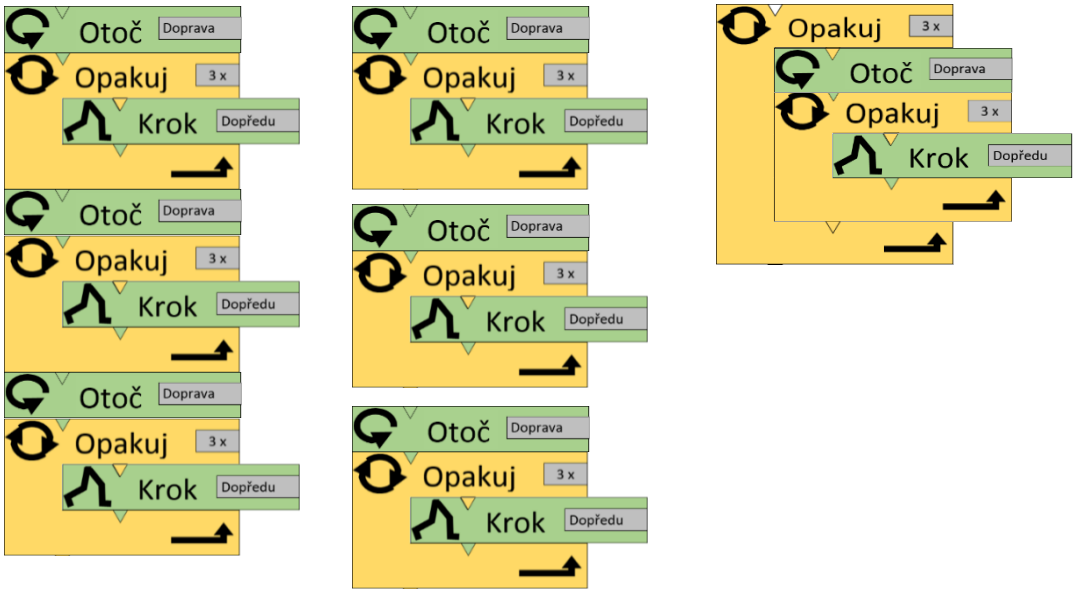
Úloha č. 5 – Algoritmy s vnořenými cykly

Tato úloha je zaměřená na utváření představ žáků o využití vnořených cyklů při sestavování počítačových programů. Žáci mají v první části úlohy za úkol, stejně jako v předchozí úloze, sestavit daný algoritmus pohybu vozidla pomocí základních příkazů, poté najít v tomto algoritmu části algoritmu, které se opakují, oddělit je od sebe a následně pomocí opakování daný algoritmus upravit (Obr. 14). Žáci musí dát pozor na otočení vozidla na startu. Vozidlo je na startu otočené směrem vlevo záměrně kvůli tomu, aby se stejné sekvence algoritmu za sebou opakovaly.



Obr. 14 - Ukázka zadání a postupu při řešení první části úlohy s vnořenými cykly

Žáci by měli sami objevit, že se ve výsledném algoritmu opět nějaká skupina příkazů opakuje. Žáci mají v druhé části úlohy za úkol najít v sestaveném algoritmu s cykly opět části, které se opakují, oddělit je od sebe a pro tyto oddělené části algoritmu znovu použít opakovací příkaz (Obr. 15). Žáci by si měli při této aktivitě uvědomit, že je možné sestavovat algoritmus tak, aby jeden cyklus obsahoval jiný vnořený cyklus.

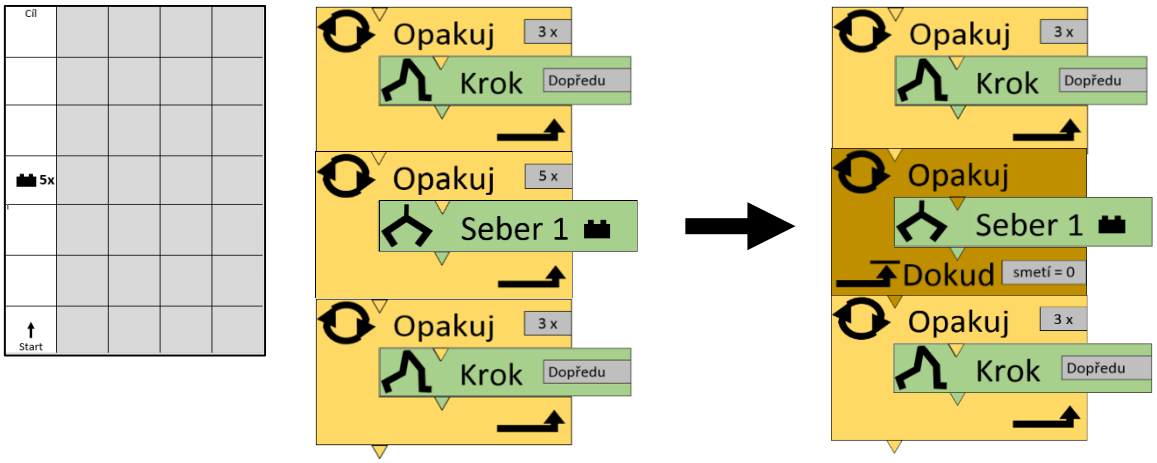


Obr. 15 - Ukázka zadání a postupu při řešení druhé části úlohy s vnořenými cykly

Žáci mají nakonec úlohy opět za úkol porovnat počet kartiček s příkazy, které použili před a po použití vnořeného cyklu. Měli by zjistit, že nejprve použijí 12 bloků se základními příkazy, a s použitím vnořeného cyklu pouze 4 bloky.

Úloha č. 6 – Algoritmy s cyklem *Opakuj – Dokud*

Pro utváření představ o smyslu a principu použití cyklu „*Opakuj – dokud*“ je důležité předchozí pochopení podstaty opakování a vnořených cyklů, na které byly zaměřené úlohy č. 4 a č. 5. Důležité je rovněž, aby žáci dokázali nalézt a určit omezení (podmínku), které cyklus (*Opakuj – dokud*) ukončí. V úloze č. 6 mají žáci za úkol sestavit algoritmus pro naložení většího počtu kostiček v situaci, kdy neví, jaký je jejich přesný počet. Je vhodné, aby se použila jednoduchá dráha pro pohyb vozidla, aby se žáci soustředili na algoritmus naložení kostiček, a nikoliv na pohyb vozidla. Žáci mají nejprve za úkol sestavit algoritmus naložení přesného počtu kostiček nejlépe již s použitím cyklů z předchozích úloh. Žáci si vytvářejí představy o sestavování programů s testovacími podmínkami (*Opakuj – Dokud*) tím, že postupně přepisují jednodušší algoritmy složitějšími algoritmy s použitím testovacích podmínek (Obr. 16).

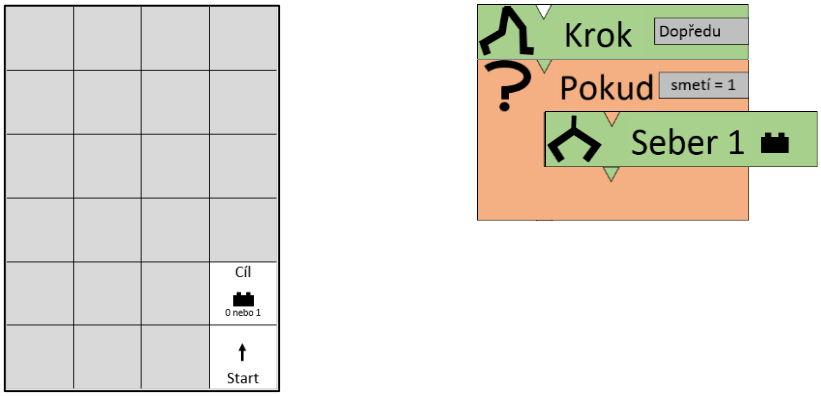


Obr. 16 - Ukázka zadání a řešení algoritmu s použitím cyklů s pevným počtem opakování a cyklem s podmínkou na konci (Opakuj – dokud)

Úloha č. 7 – Algoritmy s příkazem *Pokud*

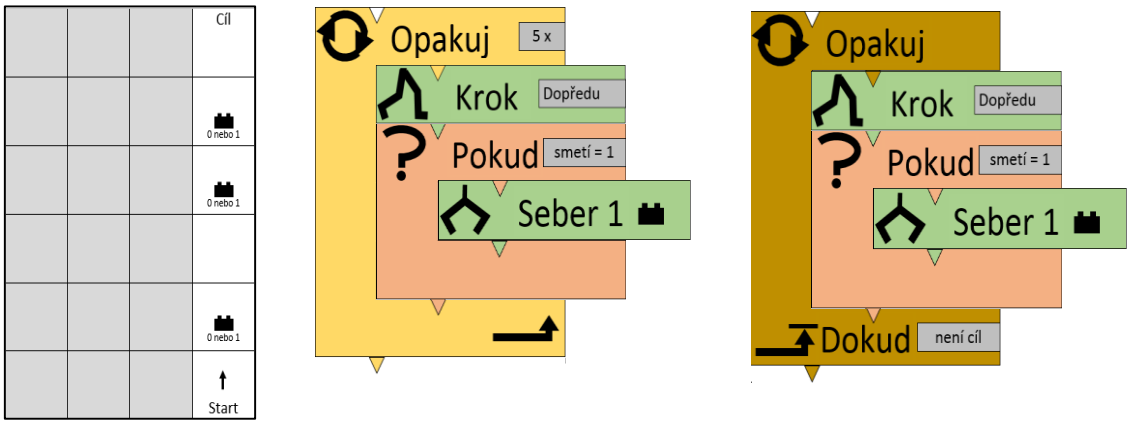
V této úloze mají žáci za úkol sestavit algoritmus pro pohyb vozidla a naložení většího počtu kostiček v situaci, kdy neví, jaký je jejich přesný počet, a zároveň neví, na které pozici dráhy pro pohyb vozidla se kostičky nacházejí. Opět je vhodné, aby se použila jednoduchá dráha pro pohyb vozidla, aby se v této úloze žáci soustředili na algoritmus naložení kostiček, nikoliv na pohyb vozidla.

Žáci mají nejprve za úkol sestavit algoritmus naložení jedné kostičky za předpokladu, že neví, jestli se tato kostička na dráze nachází (Obr. 17). Žáci by si při sestavování algoritmu měli uvědomit, že musí naprogramovat vozidlo tak, aby si na každém kroku pohybu testovalo, zda se na dráze nachází kostička, a pokud ano, aby ji vozidlo sebralo. Učitel žákům vysvětlí, že jim pro takovýto algoritmus nestačí příkazy, které dosud používali, a že musí použít nový příkaz „*Pokud*“. Po předchozí aktivitě by žáci měli sami objevit funkční princip použití příkazu „*Pokud*“. Pokud je splněná podmínka příkazu, provede se akce definovaná v těle příkazu, eventuálně pokud podmínka příkazu splněná není, neprovede se žádná akce a program pokračuje následujícími příkazy v sestaveném algoritmu. Velmi důležitým faktorem pro pochopení principu testovací podmínky „*Pokud*“ je manuální přehrávání sestaveného algoritmu. Při této aktivitě si žáci lépe uvědomují, které události nastávají v jednotlivých krocích programu.



Obr. 17 - Ukázka zadání a řešení algoritmu s použitím testovací podmínky (Pokud) pro dráhu pouze s jedním polem

Učitel žákům vysvětlí, že takto sestavený algoritmus je funkční pouze pro jedno pole dráhy (jeden krok vozidla). Žáci by si měli uvědomit, že pokud by byla dráha delší, musí tento algoritmus opakovat pro každý krok pohybu vozidla (políčko dráhy). Za předpokladu, že znají délku dráhy (počet polí dráhy), mohou žáci použít opakovací příkazy „*Opakuj Xkrát*“, nebo příkaz „*Opakuj – Dokud*“ *(Obr. 18)*. Pokud žáci použijí cyklus „*Opakuj – Dokud*“, je důležité, aby dokázali nastavit správné omezení, které tento cyklus ukončí. Za předpokladu, že žáci neznají velikost dráhy (počet polí dráhy), musí použít opakovací příkaz „*Opakuj – Dokud*“, jelikož v tomto případě nemohou v programu uvést přesný počet opakování.

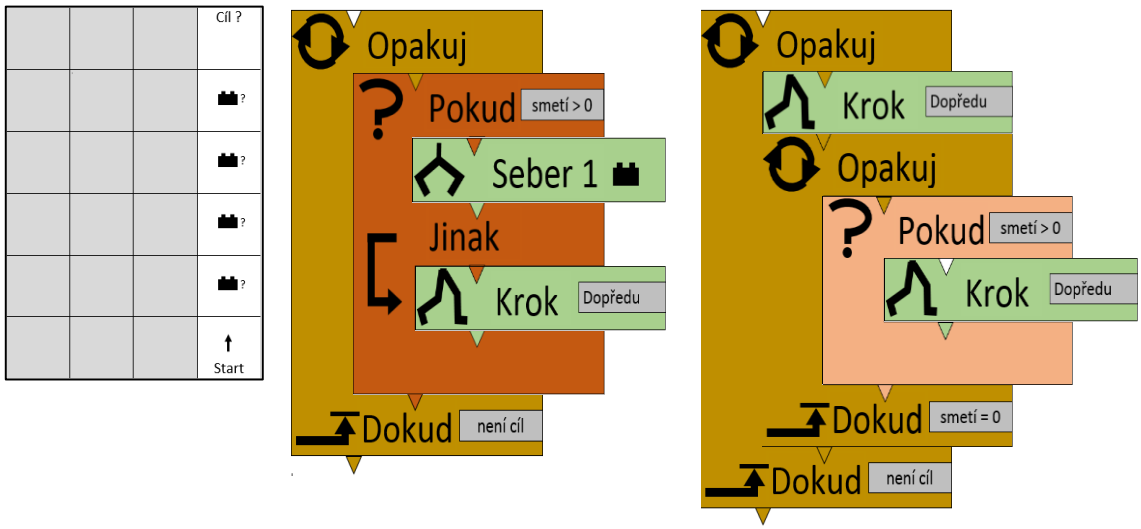


Obr. 18 - Ukázka zadání a řešení daného algoritmu s použitím cyklu s pevným počtem opakování a s použitím příkazu *Opakuj – Dokud*

Úloha č. 8 – Algoritmy s příkazem *Pokud - Jinak*

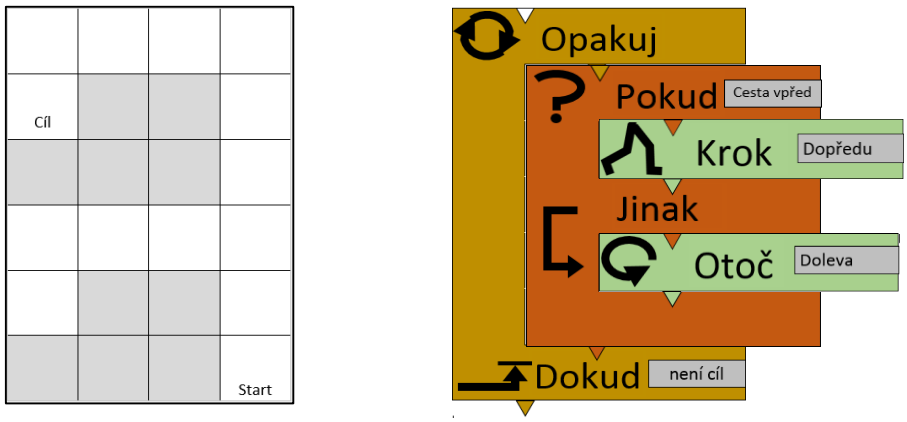
V této mají žáci za úkol sestavit algoritmus pro automatického sběrače odpadků, který se pohybuje po přímočaré dráze, protože je nutné, aby se v této úloze žáci nesoustředili na pohyb vozidla, ale na algoritmus naložení libovolného počtu kostiček v situaci, kdy stroj neví, na které pozici dráhy pro pohyb vozidla se kostičky nacházejí, a také nezná délku dráhy (Obr. 19).

Nejprve se učitel žáků zeptá, proč není algoritmus z předchozího úkolu funkční pro tento úkol (Obr. 18). Žáci by měli z algoritmu předchozí úlohy vyčíst, že algoritmus předchozího úkolu je funkční pouze za předpokladu, že se na dráze bude vyskytovat vždy pouze jedna kostička, jelikož stroj je naprogramován tak, že pokud sebere jednu kostičku, udělá následně vždy krok dopředu, a tím pádem již netestuje, jestli se na původním místě nachází další kostička. Učitel žákům vysvětlí, že tento problém se dá vyřešit použitím příkazu „*Pokud – Jinak*“, pomocí kterého mohou vozidlo naprogramovat tak, že pokud je testovací podmínka uvnitř příkazu splněna, provede nějakou činnost (Seber 1 kostičku), a pokud podmínka uvnitř příkazu splněna není, provede jinou činnost (Krok). Pro pochopení principu použití příkazu Pokud – Jinak učitel ukáže žákům funkční algoritmus bez použití tohoto příkazu (Obr. 19).



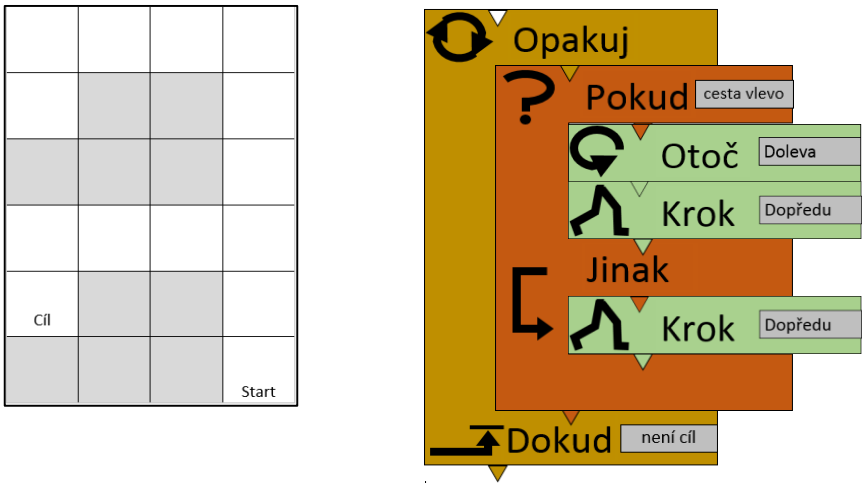
Obr. 19 - Ukázka zadání a řešení daného algoritmu s použitím příkazu *Pokud - Jinak* a bez jeho použití

Rozšiřující aktivitu s použitím příkazu *Pokud - Jinak* tvoří dvě doplňkové úlohy zaměřené na pohyb vozidla na dráze s křižovatkou. Jedná se o pohyb vozidla po stejné dráze s křižovatkou, kdy „Start“ je na stejném místě u obou dvou úloh, ale „Cíl“ je u jedné úlohy jinde než v případě druhé úlohy (Obr. 20 a 21). V první doplňkové úloze žáci sestavují algoritmus pro dráhu, kdy se má vozidlo na křižovatce pohybovat rovně (Obr. 20). Učitel nechá žáky nejdříve pracovat bez nápovědy. Pokud mají žáci se sestavením algoritmu dané úlohy problém, což se dá očekávat, zejména pak u mladších žáků, učitel žákům napoví, ať si vozidlo hlídá, jestli je volná cesta vpřed. Pokud ano, ať vozidlo „něco“ udělá, a pokud ne, ať vozidlo udělá něco jiného.



Obr. 20 - Ukázka zadání a řešení první doplňkové úlohy s použitím příkazu *Pokud - Jinak*

Ve druhé doplňkové úloze má vozidlo na křižovatce zahnout doleva (Obr. 21). Učitel opět nechá žáky nejdříve pracovat bez nápovědy. Pokud mají žáci se sestavením algoritmu dané úlohy problém, což se dá opět očekávat, zejména pak u mladších žáků, učitel žákům napoví, ať si vozidlo na každém kroku hlídá, jestli není náhodou možná vesta vlevo. Pokud ano, ať vozidlo „něco“ udělá, a pokud ne, ať vozidlo udělá něco jiného. U této úlohy je důležité, aby si žáci uvědomili, že pokud bude možná cesta vlevo a vozidlo se otočí vlevo, musí za tímto příkazem „*Otoč Vlevo*“ ihned následovat příkaz „*Krok Dopředu*“, jinak by se vozidlo znovu otočilo a algoritmus by nebyl funkční. Dá se očekávat, že si žáci nutnost tohoto kroku při sestavování algoritmu neuvědomí, proto je důležité věnovat velkou pozornost tomu, aby žáci pečlivě prováděli manuální přehrávání sestaveného algoritmu, což jim pomůže odhalit chybu a odladit sestavený algoritmus.



Obr. 21 - Ukázka zadání a řešení druhé doplňkové úlohy s použitím příkazu *Pokud - Jinak*

Rozšiřující úlohy

Scénář úloh č. 1 – 8 se dá různě modifikovat, kdy žáci sami vymýšlejí nové úkoly, při kterých sestavují algoritmy s jinými cykly. Mohou rozšiřovat instrukční sadu o možnost pohybu vozidla v obou směrech (dopředu, dozadu), mohou navrhovat pohyb vozidla po složitějších dráhách, kde budou například i křižovatky, nebo mohou navrhovat algoritmy pro vyprázdnění vozidla. Pro efektivní osvojení smyslu a principu použití vybraných algoritmických konceptů je také vhodné jednotlivé dílčí úlohy simulovat v některém blokově orientovaném programovacím prostředí (např. SCRATCH), kde žáci zjišťují, že principy algoritmického myšlení jsou ve virtuálním programovacím prostředí stejné jako v navržených „unplugged“ aktivitách a že jednotlivé akce, které jsou v těchto aktivitách sestaveny pomocí kartiček s příkazy, lze naprogramovat i ve virtuálním prostředí. Velmi zajímavou doplňující úlohou k navrženým aktivitám by bylo také využití edukačních robotů, kdy jednotlivé úlohy Automatického sběrače odpadků by bylo možné realizovat například s použitím Lego Mindstorms.

4.4 Návrh úloh a aktivit ve virtuálním programovacím prostředí

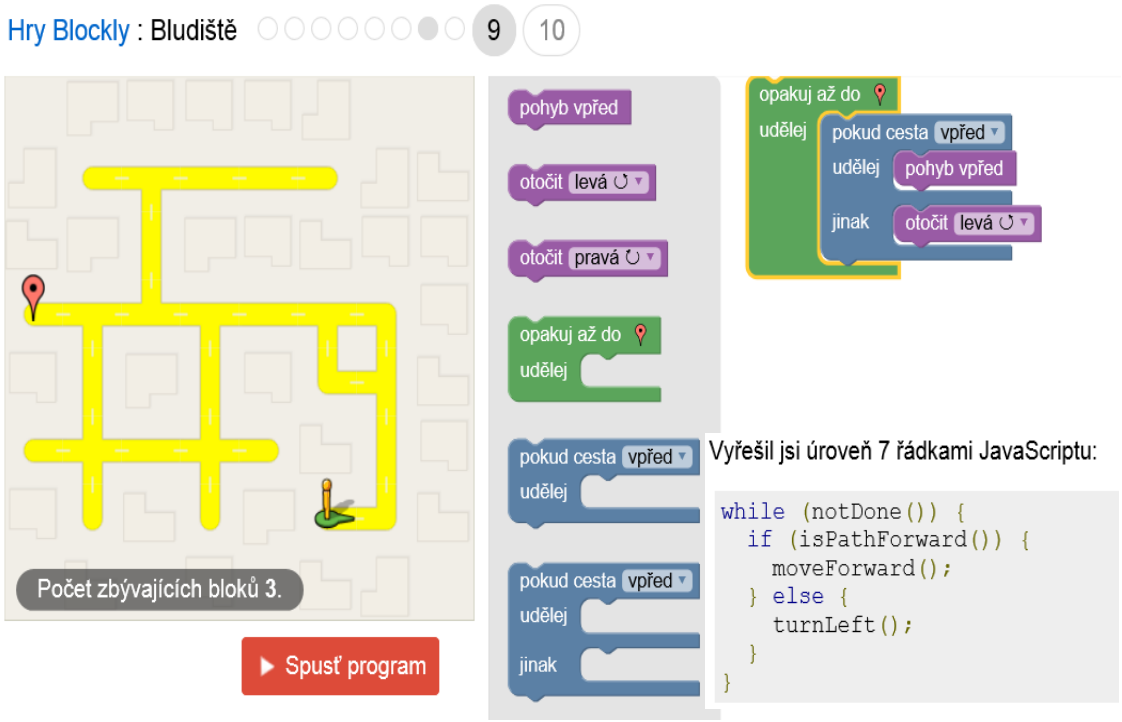
4.4.1 Dostupné počítačové aplikace a edukační programovací jazyky

Programové neboli vývojové prostředí je speciální software, který usnadňuje práci programátorům a je zpravidla zaměřen na používání konkrétního programovacího jazyka. Mezi edukační programovací prostředí, které jsou přizpůsobené a vhodné pro žáky 1. stupně

ZŠ, patří například Blockly Games, Code.org, Scratch, Logo, Baltík, Lego Mindstorms, OzoBlockly.

Blockly Games

Blockly Games je bezplatná on-line aplikace společnosti Google, ve které se nachází série vzdělávacích her, pomocí kterých se děti učí programovat. Toto prostředí lze použít v úvodních hodinách výuky programování zejména pro malé děti, které neměly předchozí zkušenosti s programováním. Jednotlivé hry jsou zaměřené na procvičení různých základních algoritmických konceptů. V tomto prostředí se žáci nemusí učit syntaxi jazyka, což bývá pro začínajícího uživatele značně demotivující. V Blockly Games žáci programují pomocí spojování už hotových příkazů a jiných prvků programu, v podobě různých dílků, bloků, ikon, které mají různý tvar nebo barvu. Tvary dílků jsou navržené tak, aby se spolu nemohly spojit dílky, které by způsobily syntaktickou chybu. Tato aplikace může mít i význam pro starší žáky při výuce programovacího jazyka JavaScript, jelikož se po úspěšném ukončení každé úlohy zobrazí řešení v tomto jazyce (Obr. 22).[28]

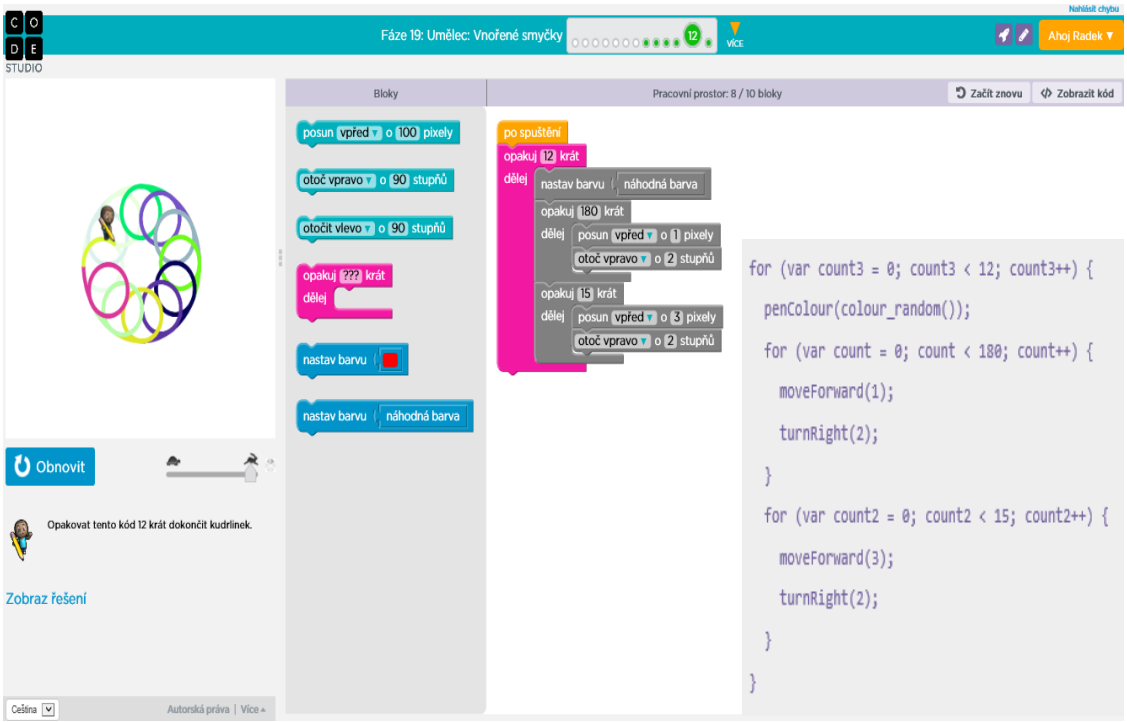


Obr. 22 – Programovací prostředí Blockly Games – Bludiště

[28] Blockly Games. Dostupné z: https://blockly-games.appspot.com/

Code.org

Code.org je bezplatná on-line aplikace, která běží od roku 2013 a je zaměřena na výuku programování pro různé věkové kategorie od mateřských škol až po střední školy. Pomocí různých kurzů, ve kterých se řeší jednotlivé úlohy, si žáci osvojují základní programovací návyky a učí se základům programování. Každý kurz se skládá z asi 20 lekcí, které víceméně odpovídají jedné výukové hodině. Jednotlivé lekce jsou zaměřené na různé algoritmické koncepty (posloupnost, cykly, vnořené cykly, podmínky, ladění programu, atd.), programování s možností vlastní kreativní tvorby nebo výuku programování pomocí aktivit bez počítače. Postupně se náročnost úloh stupňuje. Stejně jako v Blockly Games žáci programují pomocí spojování už hotových příkazů a dalších prvků programu, v podobě bloků, které mají různý tvar nebo barvu. Ve vedlejším okně si pak uživatelé mohou vyzkoušet, co tyto příkazy dělají. Po úspěšném ukončení každé úlohy si žáci také mohou zobrazit řešení v jazyce JavaScript.[29]



Obr. 23 – Programovací prostředí Code.org

[29] Dostupné z: https://studio.code.org/

Scratch

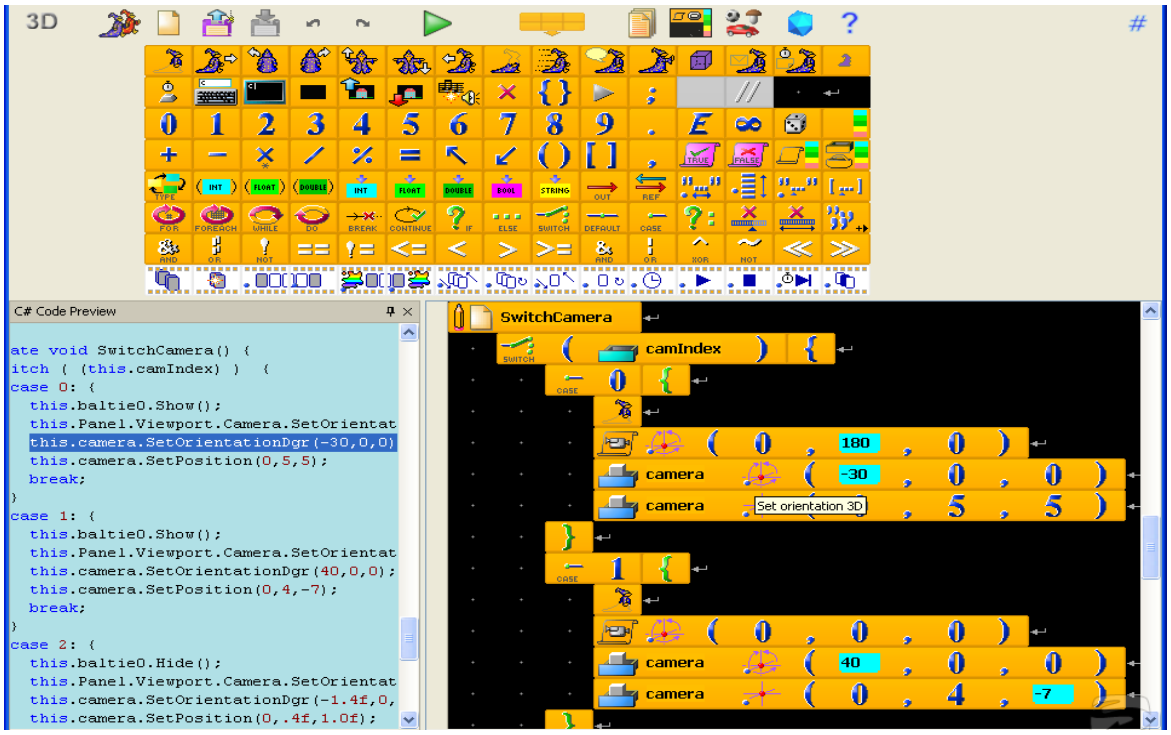
Scratch patří mezi další vizuální freewarové programovací jazyky a vývojové prostředí, u kterých se nemusí programovat v textové podobě a které umožňují vytvářet programy pomocí spojování už hotových příkazů s dalšími prvky programu v podobě různých bloků. Scratch se používá ve výuce programování už na prvním stupni základního vzdělávání a ve výuce s žáky, kteří neměli předchozí zkušenosti s programováním. Scratch lze používat zcela bezplatně a jeho obrovskou výhodou je velké množství ukázek hotových projektů a výukových materiálů, dostupných například na webových stránkách Scratch[30] nebo projektu Literacy from Scratch.[31] Ve Scratch lze naprogramovat vlastní interaktivní příběhy, hry nebo animace a následně se online podělit o své výtvory s ostatními uživateli. I přesto, že se grafické možnosti Scratch mohou na první pohled zdát velmi jednoduché, je ve skutečnosti možné naprogramovat i relativně složitou grafickou aplikaci, hru, interaktivní demo, prezentaci nebo například vizualizaci výsledků nějaké simulace.[32]

Obr. 24 – Programovací prostředí Scratch

[30] https://scratch.mit.edu/
[31] Literacy from Scratch [online]. Dostupné z: http://www.literacyfromscratch.org.uk/
[32] TIŠŇOVSKÝ, P. Scratch: plnohodnotný programovací jazyk nebo jen dětské puzzle? [online] 2011. Dostupné z: http://www.root.cz/clanky/scratch-plnohodnotny-programovaci-jazyk-nebo-jen-detske-puzzle/

Baltík

Baltík je komerční programovací a kreslící nástroj pro děti. Jedná se o vizuální programovací jazyk a vývojové prostředí určené především k výuce programování na základních a středních školách. Je založený na bázi programovacího jazyka C. Ve verzi Baltie 4 C# je možné psát i přímé textové příkazy (založen na bázi C#). Program Baltík má tři režimy – Skládat scénu, Čarovat scénu a Programovat. Na internetových stránkách společnosti SGP systems, která Baltíka vyvinula, lze najít také metodiku a výukové materiály.[33]



Obr. 25 – Programovací prostředí Baltie 4 C# Pro

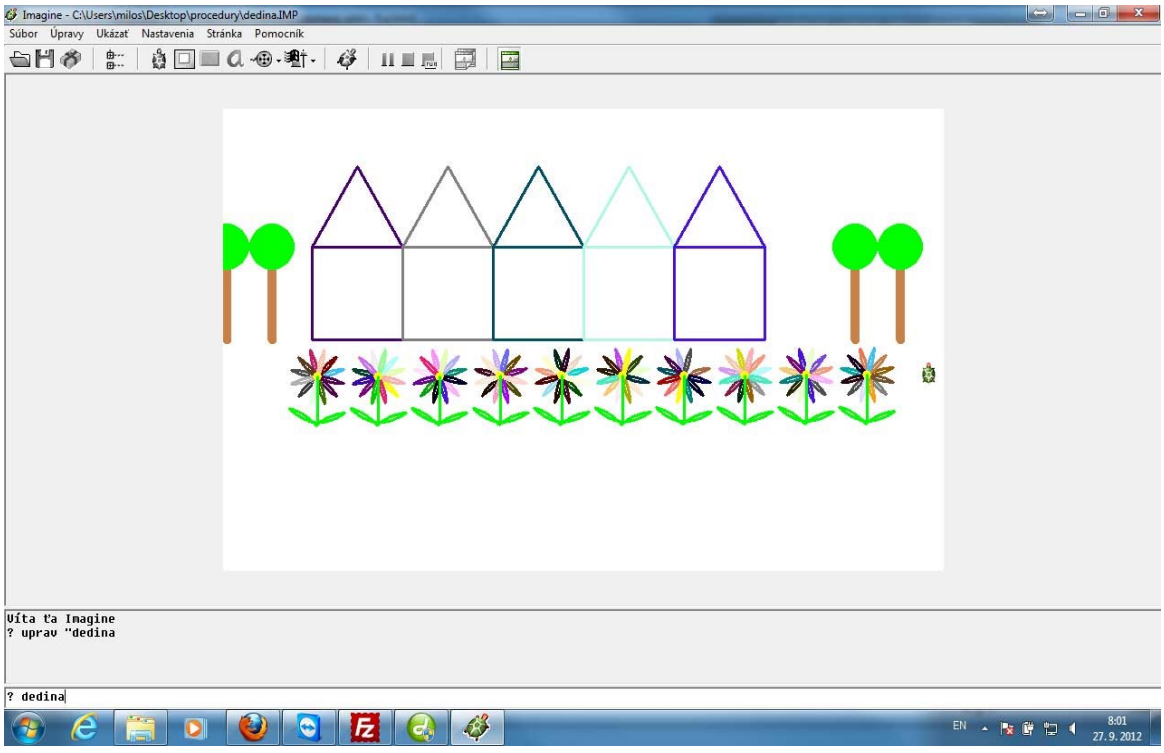
Logo

Logo je jednoduchý programovací jazyk, který vznikl v 60. letech minulého století a na jeho vývoji se podíleli S. Papert, W. Feurzlig a D. Bobrow. S. Papert očekával, že Logo přispěje ke zlepšení výuky matematiky na ZŠ a že bude pro děti představovat prostředí, v němž si budou nejen hrát, ale především rozvíjet své myšlení. Postupně vznikalo několik stovek jeho verzí. Některé z těchto implementací patří mezi komerční produkty, další jsou naopak

[33] SGP Systém - Výukové programovací nástroje C a C# pro děti, mládež i dospělé. [online] 2016. Dostupné z: http://www.sgpsys.com/cz/

nekomerční nebo šířené pod různými licencemi (freeware, různé typy open source). Jazyk Logo se stal proslulý především díky takzvané želví grafice (turtle graphics). Logo je vhodné pro podporu konstruktivního myšlení, výuku programování a v mnoha případech se jedná o první programovací jazyk, se kterým se děti ve vzdělávacím procesu setkají při řešení praktických úkolů.[34]

V českých a slovenských školách se používala verze Comenius Logo a Imagine. Imagine Logo je komerčním produktem (program byl evaluován Ministerstvem školství, mládeže a tělovýchovy), na jeho nákup lze použít dotace a je nepřímým následovníkem freewarového Comenius Loga. Má některé nové prvky, které jsou typické pro programy pod Windows, např. překrývající se grafické plochy (jako listy papíru), tlačítka i s obrázky, posuvné lišty, textová pole, lišty tlačítek apod.[35] Řadu ukázek aktivit v Imagine a výukových materiálů lze najít například na stránkách Imagine Logo – Programování pro děti.[36]



Obr. 26 – Programovací prostředí Imagine Logo

[34] TIŠŇOVSKÝ, P. Seriál Letní škola programovacího jazyka Logo [online] 2007. Dostupné z: http://www.root.cz/serialy/letni-skola-programovaciho-jazyka-logo/#ic=serial-box&icc=title
[35] Imagine [online] 2005 Dostupné z: http://imagine.input.sk/cz/popis.html
[36] Imagine logo – Programování pro děti. [online] 2011. Dostupné z: http://www.pf.jcu.cz/imagine/index.php

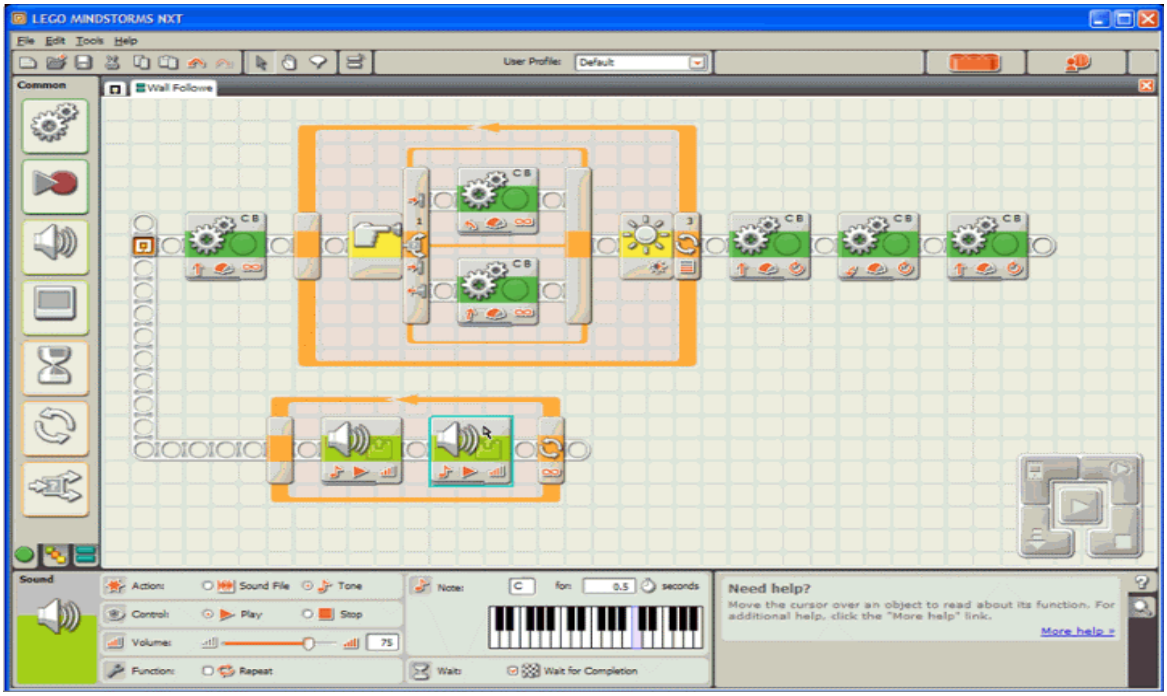
Lego Mindstorms

Edukační robotika představuje silný vzdělávací nástroj s velkým motivačním potenciálem, který umožňuje žákům prostřednictvím grafických nebo textových programovacích jazyků řídit a kontrolovat chování reálných modelů (robotů). Její edukační potenciál spočívá rovněž v možnosti zapojit žáky do řešení projektových a problémově orientovaných vzdělávacích aktivit. Při řešení projektů s použitím edukační robotiky žáci objevují technické, mechanické a obecně též přírodovědné principy a zákonitosti a rozvíjejí mezioborové znalosti. Edukační robotika představuje konstruktivistické prostředí umožňující žákům rozvíjet vlastní schopnost učit se tvorbou, poučit se z chyb a spolupracovat při řešení problémů z reálného života, které jsou založeny na použití technologií. Využívání robotů ve výuce rozvíjí kromě technických dovedností také důležité netechnické dovednosti jako je jemná motorika, vytrvalost, vizualizace, analytické myšlení nebo prostorová představivost (Tocháček, Lapeš 2012).[37]

Lego Mindstorms patří k nejvýznamnějším a nejpoužívanějším zástupcům hardwarového vybavení určeného pro realizaci aktivit z oblasti edukační robotiky. Lego Mindstorms představuje kombinaci univerzality stavebního systému Lego a dalších vyspělých technologií. Žáci oživují sestaveného robota pomocí malých servomotorů ovládaných řídící jednotkou (řídící kostkou), která se musí naprogramovat. Žáci mohou robota mimo jiné programovat v jednoduchém, intuitivním, blokově orientovaném programovacím prostředí. Aktivity s Lego Mindstorms lze implementovat do výuky programování na 1. stupni ZŠ. Robota lze programovat také pomocí programovacích jazyků, kde se instrukce zadávají v textové podobě (např. NX-C či NX-J) a vycházejí z programovacího jazyka C či Java. Navíc existuje možnost bezplatně si stáhnout aplikaci do svého tabletu nebo smartphonu, která uživatelům umožňuje zadávat svému robotovi příkazy přes inteligentní zařízení. Na stránkách Lego Mindstorms se nachází tutoriály s návody pro začátečníky, žáci zde mohou sdílet své vlastní modely nebo se nechat inspirovat výtvory ostatních uživatelů.[38]

[37] TOCHÁČEK, D., LAPEŠ, J. Edukační robotika. [online] 2012. Dostupné z: https://kraken.pedf.cuni.cz/~lapej2ap/robo/skripta_edurobo.pdf

[38] LEGO Mindstorms [online]. [cit. 2017-09-27]. Dostupné z: http://www.lego.com/cs-cz/mindstorms

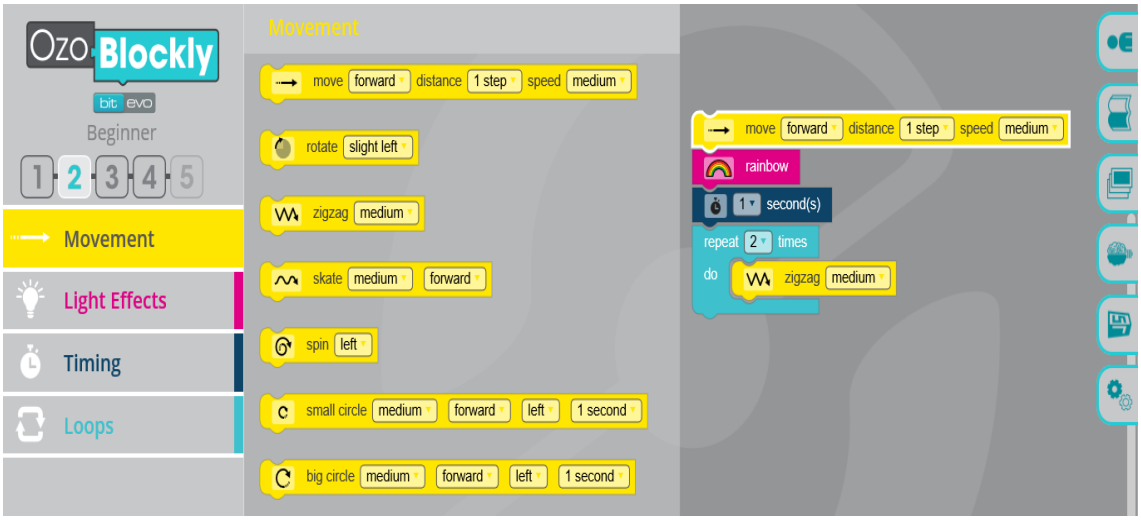


Obr. 27 – Programovací prostředí Lego Minstorms NXT

Ozoblockly

Ozobot je malý kapesní robot, který se dokáže pohybovat na různých druzích povrchu po nakreslené čáře a křivkách, reagovat na barevné kódy a měnit rychlost. Lze jej také naprogramovat pomocí počítače, chytrého telefonu či tabletu v dostupných aplikacích. V online bezplatné blokově orientované programovací aplikaci OzoBlockly se žáci učí základním programovacím návykům, základům kódování a rozvíjí si algoritmické myšlení. Žáci mohou sestavovat jednoduché programy pohybu, světelných efektů, nebo složitější programy využívající logiku, matematiku, funkce, proměnné atd.[39] OzoBlockly využívá editor Blockly společnosti Google a nabízí pět režimů, ve kterých jsou postupně zaváděny pokročilejší algoritmické koncepty. Základní úroveň „Novice“ zvládnou i menší děti, úroveň „Master“ pak dokáže plně zaměstnat i dospělé rozsáhlými řídícími funkcemi a pokročilými algoritmickými koncepty.[40]

[39] OZOBOT Getting Started - Resources and tips to get your classroom ready for Ozobot. [online] 2017. Dostupné z: http://ozobot.com/stem-education/education-getting-started
[40] OZOBLOCKLY [online] 2017. Dostupné z: https://ozoblockly.com/



Obr. 28 – Programovací prostředí OzoBlockly

4.4.2 Návrh úloh a aktivit v Code.org

Pro aktivity ve virtuálním blokově orientovaném programovacím prostředí byla vybrána bezplatná online aplikace Code.org. Tato aplikace je snadno dostupná a lze v ní najít řadu zajímavých algoritmických úloh, které se mimo jiné týkají vybraných algoritmických konceptů a svou náročností jsou vhodné i pro žáky 1. stupně základní školy. Po analýze úloh v Code.org byly vybrány úlohy, které úzce souvisejí s úlohami, které žáci řešili v rámci „unplugged“ aktivit pomocí materiálních kartiček s příkazy (Úloha č. 1 – 8). Vybrané úlohy v Code.org následně navazují na jednotlivé „unplugged“ aktivity tak, aby došlo k osvojení a upevnění získaných znalostí a dovedností ve virtuálním programovacím prostředí.

Po vytvoření účtu v této online aplikaci má učitel k dispozici nástroje pro organizaci výuky a monitorování pokroků a výsledků práce jednotlivých žáků. Je vhodné, aby žáci plnili úlohy v Code.org v modelu 1:1, kdy každý žák pracuje samostatně u jednoho počítače. Pokud není možné zajistit 1:1, existuje možnost v detailech vybraného kurzu nastavit párové programování. Efektivním nástrojem pro monitorování průběhu a výsledků práce žáků je možnost online zobrazení pokroku jednotlivých žáků v kurzu, zobrazení sestavených algoritmů žáky pro jednotlivé úlohy, počet úloh, které žáci stihli splnit ve výuce a také úspěšnost řešení jednotlivých úloh. Pomocí tohoto nástroje lze také jednoduše porovnávat pokrok a efektivitu sestavených algoritmů mezi jednotlivými žáky ve třídách, mezi třídami a ročníky.

V Tabulce č. 3 jsou přehledně představeny vybrané úlohy v Code.org a jejich návaznost na „unplugged“ aktivity, při kterých žáci řešili úlohy pomocí materiálních kartiček s příkazy (Úloha č. 1 – 8).

Tabulka č. 3 - Vybrané úlohy v Code.org a jejich návaznost na unplugged aktivity

Unplugged aktivity	**Aktivity v Code.org**
Úloha č. 1 – Posloupnost základních příkazů Úloha č. 2 – Čtení algoritmu Úloha č. 3 – Ladění algoritmu	Kurz 2, fáze 3 – posloupnost Kurz 2, fáze 10 – ladění
Úloha č. 4 – Algoritmy s cyklem *Opakuj Xkrát* Úloha č. 5 – Algoritmy s vnořenými cykly	Kurz 2, fáze 6 – cykly
Úloha č. 6 – Algoritmy s cyklem *Opakuj – Dokud*	Kurz 3, fáze 12 – while cykly
Úloha č. 7 – Algoritmy s příkazem *Pokud*	Kurz 2, fáze 17 – vytvoř příběh
Úloha č. 8 – Algoritmy s příkazem *Pokud - Jinak*	Kurz 3, fáze 8 – podmínky

Navržené aktivity v Code.org se zaměřují na procvičení a automatizaci použití příkazů s testovacími podmínkami při sestavování algoritmů a měli by žákům v budoucnu pomoci s programováním v programovacím prostředí, ve kterých mohou něco sami tvořit (Scratch, Logo, ev. Ozoblockly, Lego Mindstorms). Smyslem výuky programování nemělo být plnění konkrétních úloh, ale samotný návrh řešení a řešení obecnějších, a zejména reálných úloh, korespondujících s využitím automatizace a robotizace v běžném životě.

5 Výzkumná část

Hlavním cílem výzkumné části diplomové práce bylo naplánovat, realizovat a ověřit výuku vybraných algoritmických konceptů, opírající se o navržené „unplugged“ aktivity a aktivity ve vhodné volně dostupné počítačové aplikaci, na 1. stupni základní školy se žáky 3. – 5. ročníků. Ověřování funkčnosti navržených metodických postupů bude probíhat pomocí analýzy nashromážděných dat zachycujících postupy, chování a výsledky žáků při řešení úloh navržených „unplugged“ aktivit, aktivit ve virtuálním blokově orientovaném programovacím prostředí a při testování pochopení smyslu a principu použití vybraných algoritmických konceptů. Diplomová práce bude na základě nashromážděných dat zkoumat, do jaké míry žáci správně porozuměli a osvojili si vybrané algoritmické koncepty, a identifikovat problémy, se kterými se žáci při osvojování těchto algoritmických konceptů potýkali.

5.1 Charakteristika instituce

Pedagogický experiment byl realizován na 1. stupni ZŠ v rámci výuky informatického předmětu Práce na počítači v malé vesnické škole v podhůří Krkonoš, kterou navštěvují žáci nejen ze spádových oblastí školy. Škola má méně než 100 žáků na prvním i druhém stupni. Na prvním stupni dochází kvůli nízkému počtu žáků ke spojování tříd. Škola byla založena r. 1775. V této době se vyučovalo na faře a později v budově, ze které se stal chudobinec. V roce 1875 navštěvovalo školu 160 dětí z okolních vesnic. V roce 1889 byla postavena první část dnešní školy tzv. „stará budova“. V roce 1993 byla dokončena stavba nové školní budovy, kam byl umístěn druhý stupeň.

Ve škole probíhá využitím disponibilních hodin rozšířená výuka informatických předmětů na 1. i 2. stupni. Na 1. stupni je hodinová dotace ICT předmětu Práce na počítači jedna hodina týdně v každém ročníku (1. – 5. ročník), na 2. stupni je hodinová dotace ICT předmětu Informační technologie jedna hodina týdně v každém ročníku, kromě osmého ročníku. Výuka ICT předmětů probíhá v modelu 1:1, každý žák může pracovat na jednom zařízení. Jedná se o žákovské PC stanice, notebooky a eventuálně tablety.

5.2 Charakteristika pedagogického experimentu

Pedagogický experiment se týkal žáků 3. – 5. ročníku základní školy a probíhal v rámci výuky předmětu Práce na počítači na začátku druhého pololetí. Ve školním roce 2017/2018, kdy byl pedagogický experiment realizován, byly kvůli nízkému počtu žáků ve třídách spojeny 3. (7 žáků) a 5. ročník (8 žáků) a výuka probíhala s 15 žáky, 8 dívkami a 7 chlapci. Ve čtvrtém ročníku se pedagogického experimentu zúčastnilo 16 žáků (9 dívek a 7 chlapců). Ve spojené 3. a 5. třídě byly využity služby asistenta pedagoga. V takto heterogenně orientované skupině žáků byly služby asistenta pedagoga velmi důležité, jelikož docházelo kvůli rozdílnému věku a rozdílným úrovním nejen informatických dovedností k rozdílnému tempu práce žáků. Pedagogický experiment byl realizován podle časového harmonogramu popsaného v Tabulce 4. v kapitole 5.2.5 - Předpokládaný návrh a časová dotace pedagogického experimentu

5.2.1 Víkendové hraní s úvodem do algoritmického myšlení

Při přípravě návrhu a časové dotace pedagogického experimentu byla provedena přípravná fáze prostřednictvím pilotních lekcí s vybranými žáky, kteří se výzkumného experimentu neúčastnili. Tyto pilotní lekce probíhaly v sobotu dopoledne, a tudíž nebyly podmiňovány školní docházkou. Pilotních lekcí, aby počet žáků odpovídal počtu žáků ve skupinách při pedagogickém experimentu, se účastnili 4 žáci z 3. ročníku jiné školy (3 děvčata a 1 chlapec), s minimálními zkušenosti a dovednostmi z oblasti ICT. Při pilotních lekcích mělo dojít k ověření funkčnosti navržených materiálních pomůcek, metodických postupů a ověření předpokládané časové dotace jednotlivých lekcí se zaměřením na to, které fáze výuky je nutné důkladněji procvičovat a které lze naopak eventuálně zkrátit. Pilotní lekce ještě ukázaly jeden důležitý faktor navrženého pedagogického experimentu. Do víkendových pilotních lekcí, které nebyly podmiňovány školní docházkou a probíhaly v sobotu dopoledne, nemuseli být žáci nuceni a přemlouváni, velmi ochotně se těchto lekcí zúčastňovali a po skončení těchto pilotních lekcí chtěli ve výuce nadále pokračovat.

5.2.2 „Unplugged“ aktivity

V rámci „unplugged“ aktivit žáci řešili úlohy ve skupinách, které tvořili 3 – 4 žáci. Dvě skupiny tvořili žáci 3. ročníku, 4 skupiny tvořili žáci 4. ročníku a 2 skupiny žáci 5. ročníku. Ve spojeném 3. a 5. ročníku a ve 4. ročníku tak žáci pracovali ve čtyřech skupinách. Žáci byli do jednotlivých skupin rozděleni tak, aby se v každé skupině nacházeli děvčata i chlapci, a zároveň tak, aby se v každé skupině nacházeli žáci s rozdílným prospěchem. Takto heterogenně byly jednotlivé skupiny sestaveny úmyslně pro to, aby nedocházelo k odlišnému tempu práce, aby žáci ve skupině mohli o daném problému diskutovat a obhajovat svoje názory, a také proto, aby se žáci, kteří nějaké řešení úloh nepochopili, mohli učit od svých spolužáků a vrstevníků.

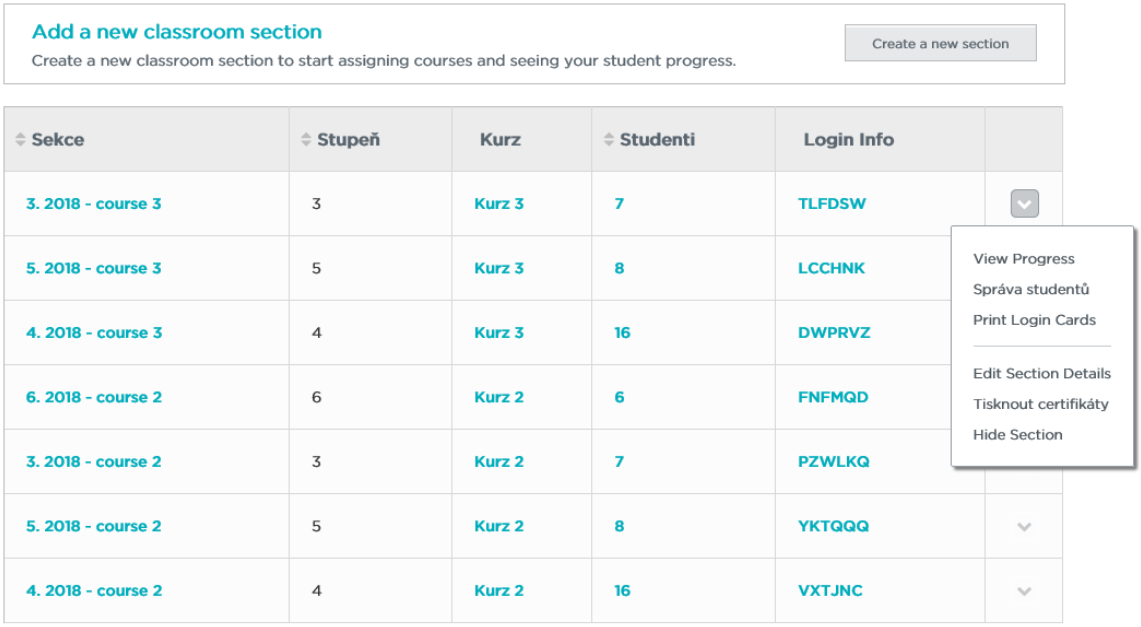
Metoda sběru dat u „unplugged“ aktivit se z velké části opírala o zkušenosti, které měly s podobnými výzkumy doc. RNDr. Naďa Vondrová, Ph.D. a Doc. RNDr. Darina Jirotková, Ph.D. z Katedry matematiky a didaktiky matematiky. Výzkumná data, týkající se práce a činností souvisejících s řešením úloh v rámci navržených „unplugged“ aktivit, byla shromažďována prostřednictvím průběžného monitorování, zapisováním poznatků z hodin a pořizováním video záznamů práce žáků během výuky. Záběry byly pořizovány pomocí smartphonu ze shora tak, aby žákům nebylo vidět do obličeje. Kamera zabírala ruce žáků a byla nastavená tak, aby došlo k zaznamenávání sestavování algoritmů jednotlivých úloh pomocí materiálních kartiček s příkazy. Tímto způsobem bylo zaznamenáváno chování, pokrok a průběh řešení jednotlivých úloh žáky a s tím i spojené eventuální obtíže.

5.2.3 Aktivity ve virtuálním programovacím prostředí

Aktivity a úlohy ve virtuálním blokově orientovaném programovacím prostření Code.org řešili žáci individuálně v modelu 1:1, kdy každý žák pracoval na jednom počítačovém zařízení. Jednalo se o žákovské PC stanice nebo notebooky. Žákům byly pro jednodušší přihlašování do Code.org vytvořeny osobní účty s hesly. Přihlašovací údaje v podobě odkazu na třídní účet a jednotlivá hesla byly vytisknuty na kartičky a rozdány žákům. Odkaz na třídní účet v Code.org byl žákům také poslán na emailovou adresu. Žáci tak mohli pro přihlášení do Code.org pouze kliknout na hypertextový odkaz, vybrat si svoje jméno a napsat svoje heslo. Tímto způsobem se žákům relativně rychle zobrazí úlohy, které mají

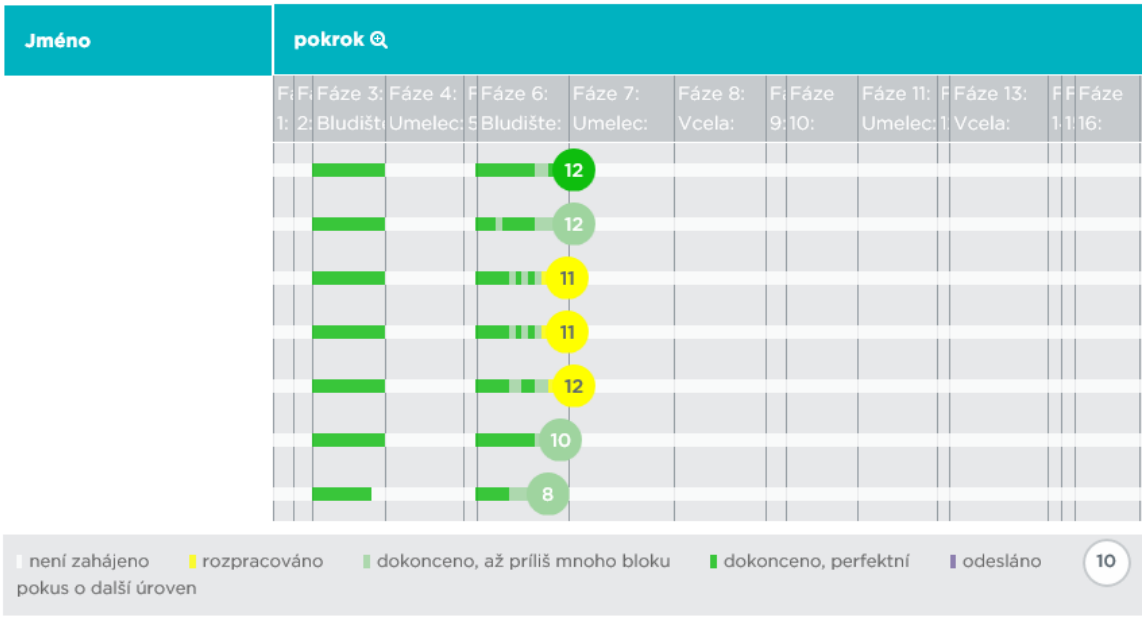
zpracovávat. Code.org také umožňuje přihlášení žáků obrázkovým heslem nebo heslem, které si žáci sami vytvoří.

Výsledky práce ve vybraném volně dostupném programovacím prostředí byly zaznamenávány online ve virtuálním blokově orientovaném programovacím prostředí Code.org. Po vytvoření účtu v této online aplikaci získal vyučující vhodné nástroje pro organizaci výuky a monitorování pokroků a výsledků práce žáků při řešení jednotlivých úloh.



Obr. 29 – Ukázka vytvořených žákovských sekcí v Code.org

Efektivním nástrojem pro monitorování průběhu a výsledků práce žáků při plnění jednotlivých úloh je pak možnost zobrazení pokroku žáků v jednotlivých sekcích. Tento nástroj umožňuje zobrazit počet a úspěšnost řešení jednotlivých úloh, které žák stihl vyřešit při výuce. Pomocí tohoto nástroje lze také jednoduše porovnávat pokrok a efektivitu sestavených algoritmů mezi jednotlivými žáky ve třídách, mezi třídami nebo mezi jednotlivými ročníky.



Obr. 30 – Ukázka zobrazení pokroku žáků jedné sekce

Pokud žáci některé úlohy nesplnili nebo nedodrželi daný počet bloků s příkazy, které měli při sestavování algoritmu použít, je možné si po kliknutí na konkrétní úlohu nechat online zobrazit jejich sestavený chybný algoritmus, a tím zjistit, s čím měli žáci při řešení úloh v Code.org problémy.

Obr. 31 – Ukázka chybně sestaveného algoritmu prostřednictvím online nástroje „Zobrazení pokroku“ v Code.org

5.2.4 Ověřování míry porozumění algoritmickým konceptům

Ověřování s využitím řízeného rozhovoru

Žáci byli průběžně testováni z hlediska ověřování pochopení smyslu a principu použití příkazů s testovacími podmínkami (Opakuj Xkrát, Opakuj – Dokud, Pokud, Pokud – Jinak). Testování probíhalo s týdenním zpožděním, aby bylo možné ověřovat nejenom pochopení, ale také osvojení a zapamatování si smyslu a principu použití jednotlivých příkazů. Žáci odpovídali na připravené otázky metodou řízeného rozhovoru s učitelem a jejich odpovědi byly zaznamenávány pomocí audio nahrávek. Tento způsob byl zvolen záměrně, jelikož jazykové a čtenářské dovednosti žáků nejen na 1. stupni ZŠ nebývají na takové úrovni, abychom si byli jisti jednoznačným pochopením zadané otázky. Otázky byly formulovány tak, aby se co nejvíce podobaly jazyku, který mladší žáci běžně používají a aby došlo k co nejlepšímu pochopení samotného smyslu jednotlivých otázek. Z každého ročníku byli postupně testováni čtyři náhodně vybraní žáci.

Ověřování s využitím praktických úloh

Porozumění žáků vybraným algoritmickým konceptům se zjišťovalo také prakticky. Žáci měli přepsat nějakou konkrétní větu z mateřského jazyka, která se týká běžné činnosti v reálném životě a má charakter algoritmu, do blokově orientovaného programovacího jazyka. Žáci při této aktivitě používali materiální pomůcky v podobě kartiček s příkazy. Příkazy a testovací podmínky, které jim chyběly, žáci dopisovali pomocí smazatelných fixů na prázdné kartičky, které tvarem odpovídaly základním příkazům a testovacím podmínkám jednotlivých příkazů. Po té se žáci snažili objevit nějaké jiné příklady z běžného života, které se podobají vybraným algoritmickým konstrukcím, a zapisovat je stejným způsobem pomocí kartiček s příkazy. Tímto způsobem žáci sestavovali vlastní bloky příkazů a zapisovali běžné činnosti z každodenního života do „řeči počítače“. Vytváření vlastních bloku příkazů, ve kterém žáci překládají rodný jazyk do programovacího jazyku srozumitelného pro stroje, dává žákům prostor, aby o jednotlivých příkazech a o smyslu a principu použití těchto příkazů při sestavování algoritmů přemýšleli sami. Výstupy a výsledky těchto „unplugged“ aktivit byly zaznamenávány pomocí fotodokumentace a jejich ukázky jsou v kapitole 6.4.

5.2.5 Předpokládaný návrh a časová dotace pedagogického experimentu

Pedagogický experiment byl navržen tak, aby na sebe navržené „unplugged“ aktivity a aktivity ve vybraném virtuálním blokově orientovaném programovacím prostředí, které se týkaly téhož vybraného algoritmického konceptu (podmínky, cykly), bezprostředně navazovaly. S výjimkou první úvodní motivační hodiny s Ozobotem, která trvala jednu vyučovací hodinu (45 minut), počítal původní návrh pedagogického experimentu s tím, že bude výuka probíhat v blocích, které budou zahrnovat tři vyučovací hodiny, mezi kterými měli žáci desetiminutovou přestávku (viz Tabulka 4). Takto byl původní návrh koncipován kvůli tomu, aby na sebe tři hodiny se stejným zaměřením navazovaly a aby mezi nimi nedocházelo k větší časové prodlevě. První hodina tříhodinového bloku byla zaměřena na pochopení smyslu a principu použití vybraných algoritmických konceptů pomocí „unplugged“ aktivit a následného manuálního přehrávání sestaveného algoritmu. Druhou hodinu žáci plnili úlohy, které tematicky navazovaly na „unplugged“ aktivity, ve virtuálním blokově orientovaném programovacím prostředí Code.org, a třetí hodina byla věnována dokončení úloh a reflexi sestavených algoritmů v Code.org. Po těchto aktivitách následovalo ověřování pochopení a porozumění žáků vybraným algoritmickým konceptům. Žáci přepisovali konkrétní příklady z reálného života formulované v mateřském jazyce s využitím vybraných algoritmických konceptů do blokově orientovaného programovacího jazyka pomocí materiálních kartiček s příkazy. Žáci měli ukázat další příklady z běžného života, které se týkají vybraných algoritmických konceptů a následně je stejným způsobem zapisovat pomocí materiálních kartiček s příkazy.

Při přípravě předpokládaného návrhu a časové dotace pedagogického experimentu se ukázala jako velice důležitá přípravná fáze v podobě pilotních lekcí. Po pilotních lekcích bylo zřejmé, kolik „unplugged“ úloh a úloh ve virtuálním blokově orientovaném prostředí Code.org jsou žáci schopni splnit během jedné vyučovací hodiny (45 minut). Na základě tohoto zjištění byl následně upraven počet a časová dotace jednotlivých úloh, zejména pak úloh zaměřujících se na výuku příkazů Pokud a Pokud – Jinak.

Tabulka č. 4 – Struktura a časová dotace navrženého pedagogického experimentu

Úvod	**1. hodina**	**Motivační hodina s Ozoboty**
1. Blok	**2. hodina**	**„Unplugged“ aktivity** • **Úloha č. 1** – Posloupnost základních příkazů • **Úloha č. 2** - Čtení algoritmu • **Úloha č. 3** – Ladění algoritmu
	3. hodina	**Aktivity v Code.org – posloupnost příkazů** • Code.org – Kurz 2, fáze 3 https://studio.code.org/s/course2/stage/3/puzzle/1
	4. hodina	**Aktivity v Code.org – ladění algoritmu** • Code.org – Kurz 2, fáze 10 https://studio.code.org/s/course2/stage/10/puzzle/1 **Dokončení a reflexe sestavených algoritmů v Code.org**
2. Blok	**5. hodina**	**„Unplugged“ aktivity** • **Úloha č. 4** – Algoritmy s cyklem ***Opakuj Xkrát*** • **Úloha č. 5** – Algoritmy s vnořenými cykly
	6. hodina	**Aktivity v Code.org** • Code.org – Kurz 2, fáze 6 https://studio.code.org/s/course2/stage/6/puzzle/1
	7. hodina	**Dokončení a reflexe sestavených algoritmů v Code.org** **Testování -** Tvorba vlastních bloků příkazů - přepis činností běžného života pomocí kartiček s příkazy ***Opakuj Xkrát***
3. Blok	**8. hodina**	**„Unplugged“ aktivity** • **Úloha č. 6** – Algoritmy s cyklem ***Opakuj – Dokud***
	9. hodina	**Aktivity v Code.org -** • Code.org – Kurz 3, fáze 12 https://studio.code.org/s/course3/stage/12/puzzle/1
	10. hodina	**Dokončení a reflexe sestavených algoritmů v Code.org** **Testování -** Tvorba vlastních bloků příkazů - přepis činností běžného života pomocí kartiček s příkazy ***Opakuj – Dokud***

4. Blok	**11. hodina**	**„Unplugged“ aktivity** • **Úloha č. 7** – Algoritmy s příkazem ***Pokud***
	12. hodina	**Aktivity v Code.org** • Code.org – Kurz 2, fáze 17 https://studio.code.org/s/course2/stage/17/puzzle/1
	13. hodina	**Dokončení a reflexe sestavených algoritmů v Code.org** **Testování -** Tvorba vlastních bloků příkazů - přepis činností běžného života pomocí kartiček s příkazy ***Pokud***
5. Blok	**14. hodina**	**„Unplugged“ aktivity** • **Úloha č. 8** – Algoritmy s příkazem ***Pokud - Jinak***
	15. hodina	**Aktivity v Code.org** • Code.org – Kurz 3, fáze 8 https://studio.code.org/s/course3/stage/8/puzzle/1
	16. hodina	**Dokončení a reflexe sestavených algoritmů v Code.org** **Testování -** Tvorba vlastních bloků příkazů - přepis činností běžného života pomocí kartiček s příkazy ***Pokud - Jinak***

Původní návrh počítal s tím, že „unplugged“ aktivity v podobě úloh zaměřených na výuku příkazů Pokud a Pokud – Jinak budou zařazeny do jedné vyučovací hodiny. Po dokončení prvotního ověření navržené metodiky v podobě pilotních lekcí bylo zřejmé, že pro lepší pochopení smyslu a principu použití těchto dvou příkazů při sestavování algoritmů je vhodné zařadit do výuky více úloh, které pak ale není možné se všemi žáky splnit během jedné vyučovací hodiny, a je tudíž nutné od sebe výuku těchto dvou příkazů oddělit. Při plnění navržených úloh opírajících se o „unplugged“ aktivity s materiálními kartičkami představujících jednotlivé příkazy je totiž stěžejní následné manuální přehrávání sestavených algoritmů. Při této činnosti žáci ve skupinách obhajují a vysvětlují své postupy. Žáci si tak lépe uvědomují, které činnosti provádí „stroj“ v jednotlivých krocích programu, následně odhalují eventuální chyby a snaží se odladit sestavený algoritmus. Na tuto činnost je nutné klást důraz a nechat žákům dostatečně dlouhý časový prostor pro správné vyřešení daných úloh. Výuka příkazů *Pokud* a *Pokud - Jinak* pak v samotném pedagogickém

experimentu bude probíhat separovaně a s časovou dotací jedna hodina týdně, jelikož struktura výuky ve škole, kde bude pedagogický experiment realizován, již nedovoluje, aby hodiny v rámci navýšené časové dotace pedagogického experimentu nadále probíhaly v tříhodinových blocích. Po dokončení a analýze výsledků pedagogického experimentu se však ukázalo, že na výsledky práce žáků takto rozdělená výuka neměla víceméně žádný vliv a výuku založenou na navržených metodických postupech lze bez problémů aplikovat při výuce informatických předmětů na 1. stupni základních škol s časovou dotací jedna hodina týdně. Výhoda blokově orientované výuky pak spočívá v tom, že pokud má některá skupina žáků při řešení úloh v rámci navržených „unplugged" aktivit nějaké problémy a nestihne do konce vyučovací hodiny algoritmus sestavit, může si tato skupina přes přestávku nechat na stole rozestavěný algoritmus a vrátit se k němu až po přestávce, což při výuce s časovou dotací jedna hodina týdně není možné. Totéž podobně platí i pro aktivity ve virtuálním programovacím prostředí.

Při pilotních lekcích byly také zjištěny první dílčí problémy při plnění jednotlivých úloh. Jednalo se například o problémy s pravolevou orientací při sestavování algoritmů a jejich následného nedůsledného manuálního přehrávání pomocí vozidla z lega. Tento drobný zádrhel se při pedagogickém experimentu u úloh v Code.org podařilo odstranit tím, že se žáci na otočných židlích natáčeli stejným směrem jako virtuální vozidlo nebo jiná virtuální postava. Ukázalo se také, že pokud někteří žáci měli problémy se sestavováním algoritmu u nějaké konkrétní úlohy, bylo přinejmenším vhodné těmto žákům napovědět, aby si vzpomněli, jak podobný příklad řešili v rámci „unplugged aktivit" s materiálními kartičkami s příkazy, a tím vytvořit jakýsi pomyslný most mezi reálnou situací a počítačovým programováním. Proto bylo velmi důležité zaměřit se na co možná nejrychlejší zpětnou vazbu, pokud k těmto obtížím při výuce dojde. Žáci, kteří měli s nějakou úlohou v Code.org problémy, dávali to vyučujícímu najevo tím, že pokládali svůj penál na svoje PC, které bylo umístěno na stole. Vyučující tak ihned viděl, který žák potřebuje pomoci. Byly také upraveny některé navržené „unplugged" úlohy tak, aby se bezprostředně týkaly problémů, které se u žáků objevily při plnění úloh v Code.org.

6 Výsledky práce

Do navrženého pedagogického experimentu byly na úvod zařazeny aktivity, které se zdánlivě tématu diplomové práce netýkají, ale jak se ukázalo, byly tyto aktivity z hlediska funkčnosti navržených metodických postupů zvoleny vhodně. Samotné pochopení posloupnosti základních příkazů pak bylo chápáno jako nutná podmínka pro pochopení funkčního principu složitějších příkazů s testovacími podmínkami.

Ve školní praxi se často nevěnuje dostatečná pozornost motivaci žáků, proto byly do navrženého pedagogického experimentu zařazeny na úvod aktivity s Ozobotem. Tato motivační hodina naprosto splnila svůj účel, pro žáky byly aktivity zábavné, víceméně bez problémů žáci s nadšením vyřešili veškeré úkoly, doma o robotech vyprávěli, o tom, co umí, jak fungují, na jaké události reagují, a řada žáků si chtěla malého robota pořídit domů. Žáci se těšili se na další lekce a jeden žák dokonce úsměvně dodal, že mu hodina rychle utekla a že by chtěl, aby mu to utíkalo pomalu. Žáci ve skupinách při řešení úloh spolupracovali, jedna skupina žáků 3. ročníku, která neodpověděla správně na všechny otázky, neměla problémy s pochopením toho, jak se Ozobot chová, ale projevila se očekávaná nedostatečná čtenářská, respektive jazyková gramotnost v podobě špatného pochopení otázky v textové podobě. Po této úvodní hodině byli žáci schopni úměrně jejich věku relativně uspokojivě odpovědět na otázku: „Co je to programování?“, „Jak fungují automatické stroje a roboti?“

Do navrhovaného pedagogického experimentu byly zařazeny také aktivity na pochopení posloupnosti základních příkazů tak, aby žáci dokázali přesně určit, které události nastanou v jednotlivých krocích programu. Při těchto aktivitách se žáci seznamovali se způsobem práce s navrženými materiálními pomůckami v podobě kartiček představující jednotlivé příkazy. Žákům bylo názorně pomocí vozidla z Lega a manuálního přehrávání sestaveného algoritmu vyučujícím předvedeno, které akce provede robot (v tomto případě žák), který algoritmus přehrává při použití základních příkazů Krok, Otoč, Seber a jakým způsobem by si měli žáci mezi sebou sestavené algoritmy přehrávat. Žáci se v sestavených algoritmech orientovali, snadno nacházeli a opravovali chybu a uvědomovali si, co počítač dělá v jednotlivých krocích programu. Pochopení posloupnosti základních příkazů se potom následně ukázalo, jako zcela nutná podmínka pro pochopení smyslu a principu použití složitějších příkazů, které s těmito základními příkazy dále pracují.

Po analýze výsledků a aktivit v Code.org, které navazovaly na navržené „unplugged“ aktivity a které byly zaměřené také na posloupnost základních příkazů, se potvrdilo očekávání, že žákům navržené „unplugged“ aktivity pomáhají se vstupem do virtuálního blokově orientovaného programovacího prostředí, v tomto případě do uživatelského prostředí Code.org. Žáci intuitivně sestavovali algoritmy, aniž by od vyučujícího dostali jakékoliv instrukce. Žáci, kteří udělali v algoritmu nějakou chybu, dokázali tuto chybu najít a odladit.

Při těchto aktivitách měli žáci očekávané problémy s pravolevou orientací, ne ve smyslu toho, že by si pletli levou a pravou stranu, ale problém jim dělalo různé otočení vozidla z Lega nebo virtuálního objektu (avatara) a jeho následný pohyb. Tento problém vyřešila jiná „unplugged“ aktivita. Žáci se otáčeli ve stoje do směru vozidla nebo na otočných židlích při práci na počítači, a tímto způsobem si lépe uvědomovali správný směr pohybu.

6.1 Výsledky a analýza navržených „unplugged“ aktivit

Ověření vhodnosti a dopadu navržených metodických postupů při sestavování algoritmů pomocí materiálních kartiček s příkazy bylo provedeno na základě analýzy videozáznamů, zachycujících činnosti, pokrok, průběh řešení jednotlivých úloh žáky a s tím i spojené problémy během výuky. Pomocí analýzy videozáznamů „unplugged“ aktivit z hlediska funkčnosti navržených metodických postupů byla u jednotlivých úloh, zaměřených na použití příkazů s testovacími podmínkami při sestavování algoritmů, ověřována tato očekávání:

- Žáci prostřednictvím navržených „unplugged“ aktivit objeví sami nebo s malou intervencí vyučujícího smysl a princip použití složitějších příkazů při sestavování algoritmů.
- Žáci prostřednictvím takto navržených metodických postupů, zejména pak prostřednictvím následného manuálního přehrávání sestavených algoritmů, lépe pochopí, jak jednotlivé příkazy sestaveného algoritmu bude postupně a za sebou provádět nějaký automat nebo stroj (v našem případě vozidlo bez řidiče).

- Žáci si pomocí navržených „unplugged“ aktivit uvědomí, které události nastanou v každém kroku programu, dokáží tento sled událostí verbálně popsat a díky tomu dokáží snadněji v sestaveném algoritmu najít chybu a sestavený algoritmus odladit.
- Žáci si uvědomí, že i když stroje vykonávají pouze základní příkazy a pracují s omezenou instrukční sadou (omezeným počtem základních příkazů), dokáží s touto omezenou instrukční sadou dále při sestavování složitějších algoritmů pracovat pomocí dalších příkazů s testovacími podmínkami.
- Žáci budou ve skupinách navzájem spolupracovat, společně navrhovat řešení dané úlohy a obhajovat svoje názory, což přispěje k porozumění vybraných algoritmických konceptů.
- Žáci nebudou mít při plnění jednotlivých úloh větší problémy.
- Navržené „unplugged“ aktivity budou pro žáky motivující a zábavné.

6.1.1 Algoritmy s cyklem Opakuj Xkrát (Úloha č. 4 a 5)

Žáci při aktivitách objevili, že jim příkaz „*Opakuj Xkrát*“ dokáže usnadnit práci a ušetřit čas při sestavování algoritmů. Dále prostřednictvím těchto aktivit poznali, že nemusí opakovat pouze jenom jeden příkaz, ale mohou opakovat i více příkazů najednou a že mohou dokonce vícekrát opakovat nějaký jiný cyklus (vnořené cykly).

Žáci pro lepší pochopení a uvědomění si toho, že jednotlivé příkazy sestaveného algoritmu bude postupně a za sebou vnímat a provádět nějaký automat nebo stroj, manuálně přehrávali sestavené algoritmy. Jeden žák ze skupiny „četl“ za sebou jednotlivé příkazy sestaveného algoritmu a jiný žák ze skupiny za pomocí vozidla sestaveného z Lega tyto příkazy prováděl. Pro žáky 1. stupně je zřejmě kvůli nedostatečnému rozvoji abstraktního myšlení relativně těžké oprostit se od smyslového vnímání a pochopit, že příkazy sestaveného algoritmu plní stroj, který dopředu neví, co všechno má udělat, ale plní postupně pouze jednotlivé konkrétní příkazy programu. U jedné skupiny ve 3. ročníku docházelo k tomu, že se žák, který algoritmus pomocí vozidla z Lega přehrával, pohyboval správně po dané dráze, i když v sestaveném algoritmu měla skupina chybu, týkající se špatného otočení vozidla. V tomto případě bylo důležité, aby se přehrávající žák oprostil od vizuálního vnímání dané úlohy a plnil pouze striktně příkazy, které mu čte spolužák z jeho skupiny. Tato chyba byla

odstraněna tím, že si přehrávající žák zakryl oči a poslepu plnil jednotlivé příkazy a tím skupina snadno odhalila a odladila chybu v sestaveném algoritmu. Na správné manuální přehrávání sestavených algoritmů bylo proto potřeba klást velký důraz. V dalších hodinách již žáci tuto chybu při přehrávání sestavených algoritmu nedělali a snažili se důkladně a poctivě algoritmy s pomocí vozidel z Lega přehrávat.

Žáci s použitím příkazu „*Opakuj Xkrát“* neměli při sestavování algoritmů žádné problémy a správně umísťovali do příkazu počet opakování a také základní příkaz, který se opakoval. V tomto ohledu žákům výrazně pomáhal tvar kartiček s příkazy, který zamezoval vytvoření syntaktické chyby. Drobné obtíže měli žáci s tím, pokud měli opakovat více příkazů najednou, pokud se v úloze opakovali celé části algoritmu. Tyto obtíže se daly odstranit tím, že žáci daný algoritmus zapsali pouze pomocí základních příkazů a následně v sestaveném algoritmu od sebe oddělovali části, které se opakují a pro lepší vizualizaci a kontrolu dávali tyto opakující se části algoritmu vedle sebe.

Žáci při řešení úloh mezi sebou ve skupinách navzájem spolupracovali, diskutovali a obhajovali svoje názory. Výsledný algoritmus byl tudíž navržen, sestaven a pomocí následného manuálního přehrávání odzkoušen a odladěn společnými silami všech žáků ve skupině. Navržené „unplugged“ aktivity byly pro žáky motivující a zábavné. Žáky bavilo zejména hraní si na roboty při manuálním přehrávání algoritmů, kdy občas mezi nimi docházelo dokonce k rozporům, kdo bude sestavený algoritmus přehrávat. Žáci dávali najevo, že se na další hodiny těší.

6.1.2 Algoritmy s cyklem Opakuj – Dokud (Úloha č. 6)

S pochopením testovací podmínky v příkazu žáci problémy neměli, správně definovali omezení, které cyklus ukončí. Žáci také pomocí navržených aktivit poznávali, že je cyklus „*Opakuj – Dokud“* univerzálnější než cyklus „*Opakuj Xkrát“*, jelikož sami zjistili, že jeden sestavený algoritmus může být funkční pro různě dlouhé dráhy nebo pro různý počet kostiček představující smetí, umístěných na dráze. Žáci neměli problém s umístěním základního příkazu do struktury cyklu, jelikož kartičky s příkazy mají takový tvar, aby jednotlivé základní příkazy zapadaly do struktury složitějších příkazů, které s těmito základními příkazy dále pracují, a aby tudíž nebylo možné udělat syntaktickou chybu.

Žáci při těchto aktivitách měli objevit, že jim příkaz „*Opakuj – Dokud“* umožňuje sestavit algoritmus s omezeným počtem základních příkazů i v případě, kdy neví, kolikrát se bude opakovat základní příkaz. Zpočátku s tím měli drobné potíže, jelikož jim dělalo problémy oprostit se od toho, že žáci vidí v reálné situaci například kolik kostiček je na ploše umístěno a kolikrát je tudíž zapotřebí příkaz zopakovat. Žákům bylo vysvětleno, že vozidlo přece nemůže dopředu vidět, kolik se na dráze nachází kostiček představujících smetí nebo jak je dlouhá dráha. Jelikož byly úlohy týkající se výuky cyklu „*Opakuj – Dokud“* pro žáky zpočátku obtížnější (zřejmě z důvodu, že žáci nebyli schopni oprostit se od vizuálního vnímání daného problému), ukázalo se, že je velmi důležité, aby se žákům co nejrychleji pomohlo, pokud mají s řešením nějaké problémy, jelikož pokud je pro žáky úloha a její řešení obtížnější a nevědí si s ní chvíli rady, velmi rychle ztrácejí o aktivitu zájem, řešení snadno vzdají, dále už nad řešením nepřemýšlejí a začínají se nudit.

Někteří žáci měli při manuálním přehrávání algoritmu za pomoci vozidla z Lega potíže se správným „přečtením“ příkazu. Nad tímto jevem se bylo potřeba pozastavit a dbát na to, aby žáci příkaz správně „četli“ zejména proto, aby žák, který algoritmus přehrává, přesně věděl, co má udělat. Správným přečtením algoritmu si také žáci dokáží lépe uvědomit, které události nastanou v jednotlivých krocích programu.

Navržené „unplugged“ aktivity byly pro žáky nadále motivující a zábavné, avšak na první pohled bylo patrné, že počáteční nadšení z něčeho nového a netradičního již nebylo na takové úrovni jako v předchozích lekcích. Žáky nadále velmi bavilo zejména hraní si na roboty při manuálním přehrávání algoritmů a dávali najevo, že se na další hodiny těší.

6.1.3 Algoritmy s příkazem Pokud (Úloha č. 7)

Zpočátku žákům dělalo drobné problémy správně pochopit průběh programu s příkazem „*Pokud“*, jelikož se domnívali, že se jedná o cyklus. Tuto představu měli žáci nejspíše proto, že v předchozích úlohách používali při sestavování algoritmů právě cykly. Žákům s představou o průběhu programu s tímto příkazem pomohlo manuální přehrávání jednotlivých kroků algoritmu učitelem. Žáci po té sami objevili, že se příkaz *Pokud* neopakuje, a když ho chtějí opakovat, je nutné ho vložit do nějakého cyklu, v tomto případě do cyklu *Opakuj Xkrát* nebo *Opakuj – Dokud*. Žáci také sami objevili, kdy se základní příkaz v těle příkazu *Pokud* provede a kdy se neprovede, což bylo příjemně překvapující, na druhou

stranu se to možná dalo očekávat, jelikož žáci s testovací podmínkou již pracovali v předchozích úlohách, které se týkaly výuky cyklů. Jedna skupina žáků 4. ročníku dokonce použila zcela správně při sestavování algoritmu k dané úloze příkaz *Pokud – Jinak*, který byl pro ně víceméně nový, nebylo jim o tomto příkazu nic řečeno a ani ho neměli použít. Tento příkaz žáci objevili mezi ostatními příkazy, které měli k dispozici, sami si ho ze sady kartiček vyndali, přemýšleli, jak funguje, použili ho při sestavování algoritmu a upozornili učitele, že se dá daná úloha vyřešit i jinak. Tato událost byla velice potěšující a díky tomu bylo zřejmé, že žáci o průběhu programu dokáží přemýšlet.

Ukázalo se, že při utváření správných představ žáků o průběhu programu s testovacími podmínkami hraje velmi důležitou roli precizní následné manuální přehrávání sestavených algoritmů, při kterém si žáci dokáží vybavit, které události nastanou v jednotlivých krocích algoritmu, což jim následně pomáhá s rozvojem abstraktního a algoritmického myšlení. Žáci prostřednictvím takto navržených metodických postupů, zejména pak prostřednictvím následného manuálního přehrávání sestavených algoritmů, lépe pochopí, že jednotlivé příkazy sestaveného algoritmu bude postupně a za sebou vnímat a provádět nějaký automat nebo stroj, v tomto případě vozidlo bez řidiče.

Ukázalo se, že navržené „unplugged“ aktivity již přestávají být pro žáky 5. ročníku zajímavé, jako tomu bylo v předešlých úlohách. Po analýze tohoto zjištění bylo zřejmé, že žáci 5. ročníku měli dané úlohy mnohem rychleji vyřešené než žáci 3. ročníku, se kterými jsou spojeni, a doplňkové aktivity k daným úlohám už nechtěli plnit. Při doplňkových aktivitách pro rychlejší skupiny měli žáci opravit sestavený algoritmus pro pohyb vozidla po dráze v situaci, kdy se bude někde na dráze nacházet kostička představující smetí. Při těchto doplňkových úlohách se žáci 5. ročníku nejspíše nudili a začali provádět aktivity, které s řešením doplňkových úloh nesouvisely, například upravovali vozidlo z Lega. Žáci 5. ročníku u plnění doplňkových úloh dávali učiteli najevo, že už by rádi řešili úlohy v Code.org. Toto jim bohužel vzhledem ke zbývajícímu času do konce hodiny a organizaci výuky nemohlo být umožněno. Naopak aktivity byly stále zábavné a motivující pro žáky 3. a 4. ročníku, a jelikož je tyto aktivity bavily, vznikaly dokonce drobné neshody v tom, kdo bude simulovat chování robotického vozidla. Stejně jako žáci 5. ročníku dávali najevo, že se už těší na řešení úloh v Code.org.

6.1.4 Algoritmy s příkazem Pokud – Jinak (Úloha č. 8)

S pochopením smyslu a funkčního principu použití příkazu *Pokud* rozšířeného větví *Jinak* žáci víceméně problémy neměli, jelikož v předchozích aktivitách sestavovali algoritmy se samotným příkazem *Pokud.* Správně sami odhalili, že pokud bude testovací podmínka příkazu *Pokud - Jinak* splněna, provede se „první“ základní příkaz, a pokud ne, provede se „druhý“ příkaz ve struktuře větve *Jinak*, na rozdíl od příkazu *Pokud*, kdy pokud testovací podmínka splněna není, příkazy v těle se neprovedou a program pokračuje dál. Pokud žáci chtěli při sestavování algoritmu příkaz *Pokud – Jinak* opakovat, nedělalo jim problémy vnořit tento příkaz do nějakého cyklu (*Opakuj Xkrát, Opakuj – Dokud*). Drobné problémy dělalo žákům pochopení hodnot v testovací podmínce, a zejména to, ve kterých situacích nastane splnění a nesplnění testovací podmínky příkazu *Pokud – Jinak*. K odstranění tohoto problému opět pomohlo manuální přehrávání algoritmu samotným vyučujícím, který kladl důraz na to, aby žáci v jednotlivých krocích programu sami určovali, zdali je nebo není testovací podmínka příkazu splněna a které akce se provedou právě při splnění nebo nesplnění testovací podmínky příkazu *Pokud – Jinak*.

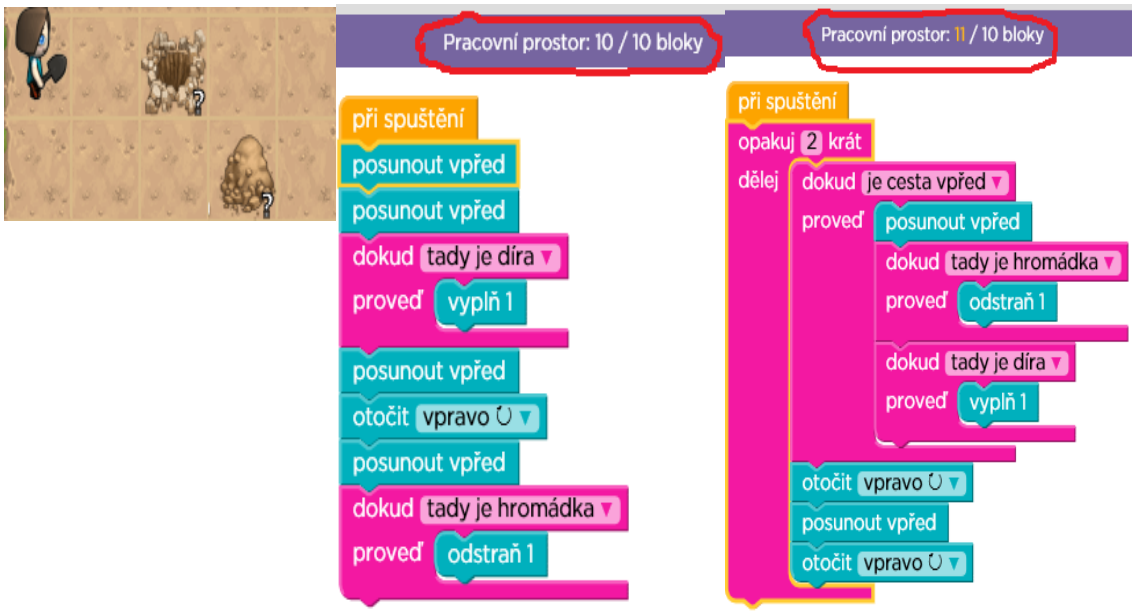
Při těchto aktivitách byl kladen velký důraz na precizní „čtení“ a manuální přehrávání sestavených algoritmů. Bylo zřejmé, že pokud žáci měli s manuálním přehráváním algoritmů nějaké problémy, nedokázali si uvědomit průběh programu a dělali při této aktivitě chyby. Žákům dokázalo výrazně pomoci manuální přehrávání sestaveného algoritmu vyučujícím, při kterém si dokázali uvědomit průběh programu, představit si, které akce se provedou při splnění či nesplnění testovací podmínky příkazu, a odhalit a odladit případnou chybu v sestaveném algoritmu.

Žáci ve skupinách při sestavování algoritmů a při manuálním přehrávání algoritmů spolupracovali, obhajovali si svoje názory, navzájem se kontrolovali a eventuálně se opravovali. Pro některé starší žáky však navržené „unplugged“ aktivity již nebyly tolik zajímavé, jelikož si snadněji a rychleji vytvořili představy o smyslu a principu použití příkazů s testovacími podmínkami, manuální přehrávání algoritmů jim už připadalo dětinské a upřednostňovali aktivity v Code.org, což bylo na jednu stranu potěšující, na druhou stranu dokázali narušovat plynulý průběh připravené výuky.

6.2 Výsledky a analýza navržených aktivit v Code.org

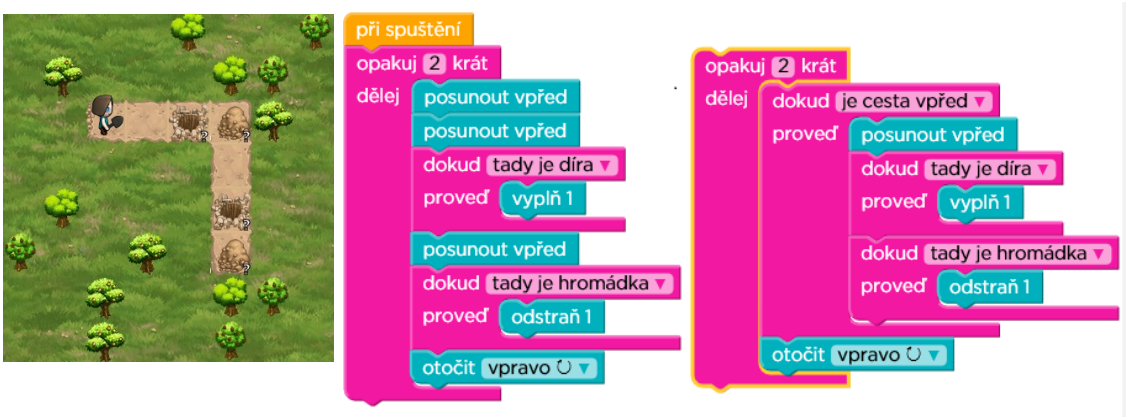
Analýza výsledků práce žáků ve virtuálním programovacím prostředí Code.org byla provedena pomocí nástrojů pro zaznamenávání pokroku žáků. Díky tomuto nástroji lze sledovat počet úloh, které žáci dokážou během výuky vyřešit, úspěšnost řešení, lze si nechat zobrazit nesprávně sestavené algoritmy nebo jednoduše porovnat pokrok mezi žáky v ročníku nebo mezi ročníky. Díky tomuto nástroji je možné zjistit například, se kterou úlohou mělo více žáků problémy, a zaměřit se na tuto úlohu podrobněji při následné reflexi sestavených algoritmů.

Z analýzy chybně sestavených algoritmů je patrné, že žáci, a zejména mladší žáci 3. ročníku, mají při sestavování algoritmů řadu problémů, které s pochopením smyslu a principu použití příkazů s testovacími podmínkami nesouvisejí a souvisejí spíše s nedodržením předepsaného počtu příkazů, který žáci měli v Code.org použít. Žákům 3. ročníku také dělalo obtíže objevit v sestaveném algoritmu pomocí základních příkazů větší počet příkazů nebo celých částí algoritmu, které se opakují. Téměř všichni žáci měli problémy s řešením jednoho algoritmického problému, kdy bylo v Code.org pro efektivní sestavení algoritmu vyžadováno přidání zdánlivě zbytečného příkazu na konec algoritmu tak, aby se dala poslední část algoritmu také opakovat. Jeden žák měl poznámku: *„Proč bych tam měl dávat příkaz navíc, když jsem úkol splnil.“* Na vyžadovaný způsob řešení v Code.org byli následně žáci upozorněni a bylo jim vysvětleno, že by se měli oprostit od toho, že stačí pouze přemístit vozidlo z bodu A do bodu B a že by se měli v tomto případě pokusit (přidáním jednoho zdánlivě zbytečného příkazu navíc) sestavit jednodušší, efektivnější, kratší a zároveň samozřejmě také obecně funkční algoritmus. Na základě tohoto zjištění byla s žáky provedena také doplňková „unplugged“ aktivita, která se tomuto jevu věnovala. Na druhou stranu není smyslem výuky programování na 1. stupni ZŠ, aby žáci řešili daný problém jediným možným způsobem, k čemuž je víceméně Code.org svým pracovním prostorem (Obr. 32) nutí. Naopak důležité je, aby si žáci během výuky rozvíjeli algoritmické myšlení, aby přemýšleli, aby se učili problém rozložit na dílčí kroky a aby mohli podle svého rozvoje abstraktního myšlení sestavit funkční algoritmus různými způsoby. V reálném životě by jistě například nebylo vhodné, aby autonomní vozidlo provádělo zbytečné pohyby jenom proto, aby mohl programátor napsat efektivnější program.



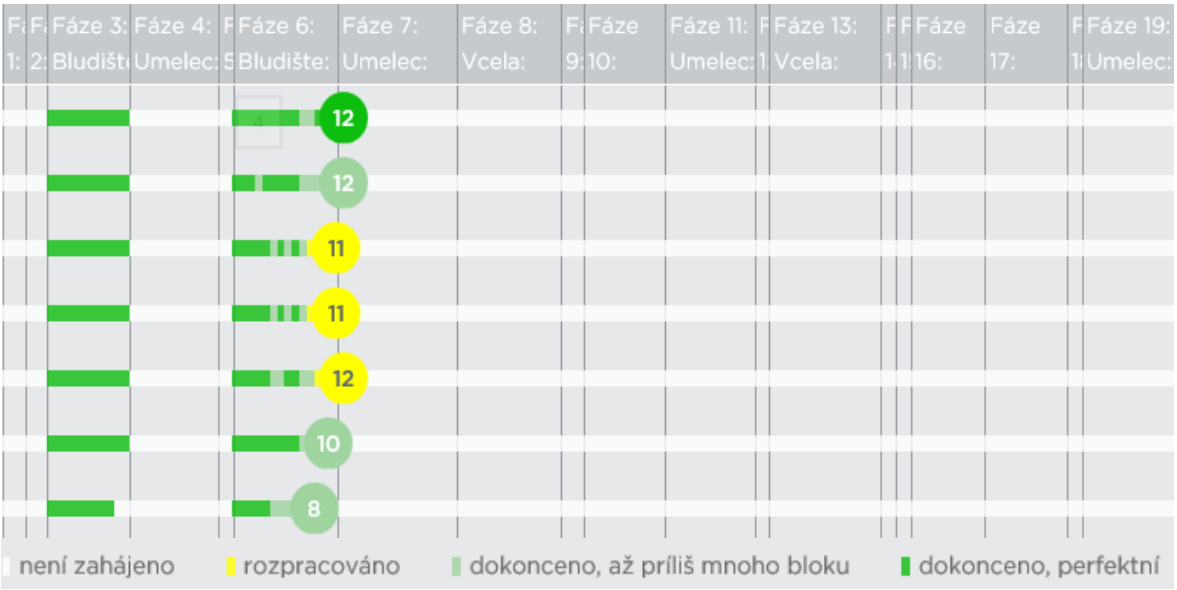
Obr. 32 – Ukázka rozdílných řešení úlohy a požadovaného „Pracovního prostoru“

Pro správný rozvoj algoritmického myšlení je důležité, aby se žáci dokázali oprostit od smyslového, v tomto případě vizuálního, vnímání, uvědomovali si, jak se chová robotický stroj, aby se vžili do situace, jak ovládat stroj, robotické zařízení, které „rozumí“ jen omezené sadě příkazů a nemá vlastnosti a schopnosti, které má člověk. V ukázce jedné úlohy (Obr. 33) by bylo možná vhodnější nechat žáky sestavit obecně funkční algoritmus pro případ, kdy se na vyznačeném poli někde nachází různě velké hromádky hlíny a jámy, kdy stroj musí být naprogramován tak, aby automaticky prošel všechna místa daného pole a na každém kroku testoval, jestli se tam nenachází hromádka hlíny, kterou by odstranil nebo jáma, kterou by zasypal. Sestavit algoritmus s více vnořenými cykly může být pro žáky 1. stupně značně složité, zejména pak pro žáky 3. ročníku. Pokud však žáci již na prvním stupni získávají takovéto zkušenosti a náhled do složitějších problémů algoritmizace, mohou tyto zkušenosti využít v pozdějším věku a lépe si poradit s náročnějšími úlohami.

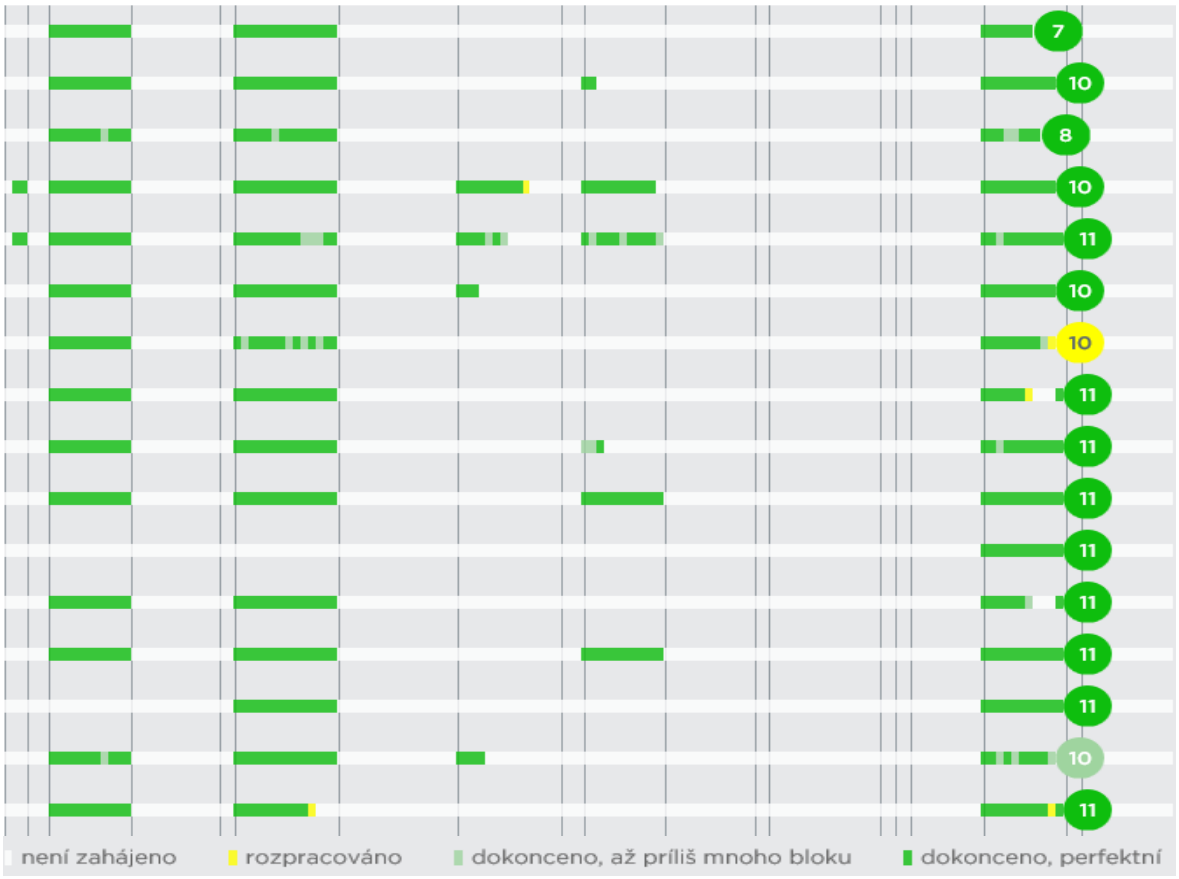


Obr. 33 – Ukázka možného řešení úlohy v Code.org - Kurz 3, fáze 12, úloha 9

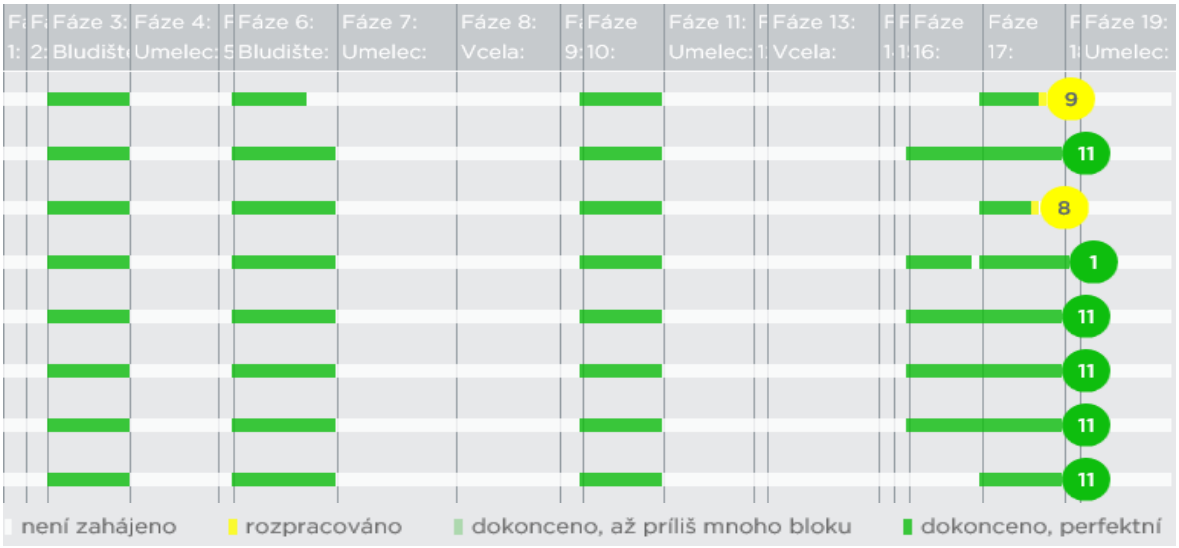
Po analýze výsledků žáků v Code.org (Obr. 34 - 36) lze konstatovat, že mezi žáky byly jen malé rozdíly v počtu a úspěšnosti řešených úloh napříč ročníky a z genderového hlediska. Tyto rozdíly byly mnohem zřetelnější mezi ročníky.



Obr. 34 - Ukázka pokroku žáků 3. ročníku v Code.org - Kurz 2



Obr. 35 - Ukázka pokroku žáků 4. ročníku v Code.org - Kurz 2



Obr. 36 - Ukázka pokroku žáků 5. ročníku v Code.org - Kurz 2

Ze zobrazeného pokroku žáků 3. – 5. ročníku v Code.org je patrné, že rozdíly v počtu a úspěšnosti řešených úloh jsou méně patrné mezi žáky 4. a 5. ročníku. Oproti nim je však

počet a úspěšnost řešených úloh menší u žáků 3. ročníku. Žákům 3. ročníku oproti žákům 4. a 5. ročníku trvalo mnohem déle přihlášení na emailový účet, kde měli hypertextový odkaz na vybrané lekce v Code.org, a také jim po kliknutí na tento odkaz zabralo více času samotné přihlášení do Code.org. Tyto rozdíly jsou samozřejmě způsobeny rozdílným věkem, tempem práce a tomu odpovídající úrovni digitální gramotnosti.

U žáků se neobjevily kromě nedokončených posledních úloh chybně vyřešené úlohy (žlutá barva). Žákům úmyslně nebylo řečeno, že v případě problémů s nějakou úlohou mohou řešit úlohy jiné z toho důvodu, aby řešení nevzdávali a snažili se úlohy dokončit. Někdy se to však neobešlo bez intervence vyučujícího.

Úspěšnost a počet úloh, které žáci dokázali vyřešit v Code.org, nesouvisí s jejich školním prospěchem. Někteří slabší žáci, kteří v jiných předmětech nebyli úspěšní, dokázali být při programování velmi pozorní a bystří a někteří se dokonce věnovali programování i mimo školu. Při programování dostávají tito žáci prostor pro svoji realizaci a mohou být ve srovnání s žáky s dobrým prospěchem při programování úspěšnější a šikovnější.

6.3 Výsledky a zjištění z řízených rozhovorů se žáky

Řízené rozhovory s vybranými žáky byly původně naplánovány na třetí hodinu vyučovacího tříhodinového bloku. Ukázalo se však, že začlenění této aktivity do vyučovací hodiny je zcela nevhodné, jelikož při řízeném rozhovoru byly kladeny otázky vždy pouze jednomu žákovi a ostatní žáci plnili úkoly v Code.org. Při aktivitách v Code.org mívají žáci mnohdy jisté obtíže různého charakteru, které bylo nutné zaznamenávat, a samozřejmě se některým žákům věnovat a poskytovat jim zpětnou vazbu. Jedním z důležitých poznatků pedagogického experimentu byla skutečnost, že pokud docházelo u jednotlivých žáků k pochopení konkrétního problému, většinou s dopomocí učitele, následující úlohy tito žáci plnili bez větších obtíží. Řízené rozhovory tudíž probíhaly separovaně a zhruba s týdenním zpožděním od navrženého pedagogického experimentu. Tímto způsobem pak docházelo k ověřování nejenom pochopení, ale také osvojení a zapamatování si smyslu a principu použití jednotlivých příkazů. Z každého ročníku byli pro řízené rozhovory vybráni náhodně minimálně čtyři žáci. Jednalo se o dva žáky (chlapce i dívky) z 3. ročníku, čtyři žáky ze 4. ročníku a dva žáky z 5. ročníku.

Žákům byly postupně kladeny jednotlivé otázky. Jelikož některé odpovědi, zejména odpovědi žáků 3. ročníku, byly víceméně zmatečné a někdy jejich odpovědi s položenou otázkou dokonce vůbec nesouvisely, byly žákům kladeny další upřesňující otázky tak, aby došlo k pochopení samotného smyslu položené otázky. Uvedené odpovědi nejsou tudíž celkovým přepisem audiozáznamů, ale jsou zde uvedeny pouze odpovědi, které s danou otázkou souvisí. Odpovědi žáků jsou uvedeny tak, jak byly zaznamenány v původním znění pomocí audionahrávek včetně hovorového jazyka (nespisovné češtiny).

Odpovědi žáků na testovací otázky

Při tvorbě otázek nebyly používány odborné výrazy a byla snaha otázky formulovat tak, aby smysl otázky pochopili i nejmladší žáci. Otázky se zaměřovaly na ověřování správného porozumění smyslu a principu použití daných příkazů při sestavování algoritmů: Proč se příkaz používá, v jakých situacích a kdy se příkaz používá a jakým způsobem se daný příkaz používá.

Otázka 1: Proč je výhodné používat příkaz *„Opakuj Xkrát“*?

3. ročník: Žák 1 – *„Protože nám to ulehčí práci.“* **Žák 2** – *„Protože by to bylo lehčí. Těch kroků by tam bylo víc.“*

4. ročník: Žák 1 – *„Abysme si usnadňovali práci, protože tam nemusíme dávat tolik těch příkazů.“* **Žák 2** – *„Je to zkratka, můžeme si to zkrátit.“* **Žák 3** – *„Bylo by toho moc. Tímhle si ulehčuju práci.“* **Žák 4** – *„Abysme to udělali jednodušejc na míň kroků.“*

5. ročník: Žák 1 – *„Zjednodušuje to práci.“* **Žák 2** – *„Protože to je kratší.“*

Otázka 2: Vysvětli, v jakých případech se dá příkaz „*Opakuj Xkrát*“ použít.

3. ročník: Žák 1 – *„Protože se tam něco opakuje.“* **Žák 2** – *„Že třeba tady dáme 3 a tady je třeba opakuj vpřed, tak že ono to bude třeba opakovat 3x, půjde 3x vpřed. Opakuj 3x krok vpřed.“*

4. ročník: Žák 1 – *„Když máme něco opakovat.“* **Žák 2** – *„Když třeba potřebujeme opakovat kroky a musíme to udělat třeba 3x.“* **Žák 3** – *„Když se něco opakuje. Třeba krok dopředu.“* **Žák 4** – *„Třeba když je rovná dráha, tak si můžeme dát jeden krok dopředu a on to bude opakovat.“*

5. ročník: Žák 1 – *„Aby jsme tam nemuseli mít hodně těch bloků.“* **Žák 2** – *„Když máme třeba furt sbírat kostičky, třeba takhle.“*

Otázka 3: Co musíš do opakovacího příkazu „*Opakuj Xkrát*“ zadat?

3. ročník: Žák 1 – *„Tady kolikrát a tady to, co má udělat.“* **Žák 2** – *„Tady sem čísla a tady třeba krok dopředu. Aby opakoval 5x krok dopředu.“*

4. ročník: Žák 1 – *„Sem se dá počet, kolikrát to chceme opakovat, a sem, co chceme opakovat. Sem se třeba dá krok dopředu a sem třeba jako 4x, tak hned všem dojde, že vozidlo má udělat 4 kroky.“* **Žák 2** – *„Kolikrát to chceme opakovat a třeba když chceme opakovat krok nebo doprava nebo doleva. Třeba opakuj 2x krok.“* **Žák 3** – *„Kolikrát to chceš opakovat a co chceš opakovat.“* **Žák 4** – *„Dá se do něj, co potřebujeme opakovat a kolikrát to potřebujeme opakovat.“*

5. ročník: Žák 1 – *„V tom malým okýnku je třeba, kolikrát to chceme opakovat, a v tom velkým to co chceme opakovat.“* **Žák 2** – *„Kolikrát se to má opakovat a co se má opakovat.“*

Otázka č 1 – 3 měla ověřit porozumění smyslu a principu použití příkazu „*Opakuj Xkrát*“. S odpověďmi na tyto otázky žáci víceméně neměli žádné problémy a nemusely jim být kladeny žádné další doplňující, upřesňující nebo vysvětlující otázky.

Otázka 4: Je možné opakovat více různých příkazů najednou? Pokuste se vysvětlit.

3. ročník: Žák 1 – *„Jo. Kdybych měla třeba dva kroky opakovat víckrát, tak by to bylo snadnější.“* **Žák 2** – *„Třeba opakovat krok, otočit vlevo, krok, otočit vpravo. Udělá schody.“*

4. ročník: Žák 1 – *„Můžeme. Podle toho, jak to potřebujeme. Může tam být jeden nebo jich tam můžeme dát víc.“* **Žák 2** – *„Je to možný. Když to chceme 2x tak dáme do toho malýho okýnka 2x a sem dáme třeba krok, doleva nebo doprava.“* **Žák 3** – *„Jo. Třeba schody. Krok dopředu, otoč vlevo, krok dopředu, otoč vpravo.“* **Žák 4** – *„Jo. Třeba opakovat 6x krok dopředu, otoč vlevo, krok dopředu, otoč vpravo.“*

5. ročník: Žák 1 – *„Jo. Třeba dvakrát opakovat jakože se posune o krok dopředu, sebere jednu, o krok dopředu a sebere jednu.“* **Žák 2** – *„Ano. Třeba 4x opakovat otoč doleva a krok dopředu. Udělal by čtverec.“*

Pomocí této otázky se mělo zjistit, zda žáci po realizované výuce pochopili, že mohou opakovat nejenom jeden základní příkaz, ale že mohou opakovat i více základních příkazů najednou. Žáci si relativně rychle vzpomněli na „unplugged“ aktivity a aktivity v Code.org, kde se s touto problematikou setkali, a jako příklad většinou dávali algoritmy použité právě v těchto aktivitách.

Otázka 5: Můžeme vložit do jednoho opakovacího příkazu jiný opakovací příkaz? Vysvětli.

3. ročník: Žák 1 – *„Jo.“* **Žák 2** – *„Můžem.“* Vysvětlit nedokázali.

4. ročník: Žák 1 – *„Jo. Tam jsem to častokrát využil na tom počítači. Jsem si dal třeba do jednoho příkazu 3x a do druhého 5x krok dopředu a sem zbytek příkazů, které jsem chtěl opakovat. Takže to udělalo 3x 5 kroků dopředu a otoč třeba doleva.“* **Žák 2** – *„Ano můžeme“* (Vysvětlit nedokázal) **Žák 3** – *„Jo. Může.“* (Vysvětlit nedokázal) **Žák 4** – *„Jo. Třeba když chcem opakovat to, co se už opakovalo.“*

5. ročník: Žák 1 – *„Jo. Třeba dvakrát opakovat tři kroky dopředu a sebrat jednu kostičku.“* **Žák 2** – *„Nevím, nepoužil jsem to.* (pozn. autora – použil to několikrát)*“*

Otázka 5 se týkala pochopení možnosti vložení jednoho opakovacího příkazu do jiného opakovacího příkazu. I když někteří žáci na otázku odpověděli správně, nedokázali uvést nějaký příklad. Při pokusu o vysvětlení však bylo patrné, že o této možnosti žáci povědomí mají. Dalšími výzkumy by bylo vhodné zjistit, proč žáci neuměli na takovou otázku uspokojivě odpovědět. Jeden žák 5. ročníku si na aktivity, kde vnořené cykly použil, nevzpomněl a neodpověděl uspokojivě.

Otázka 6: Proč jste museli použít příkaz „*Opakuj – Dokud*“ místo příkazu „*Opakuj Xkrát*“?

3. ročník: Žák 1 – *„Třeba umyj nádobí, aby bylo čistý. Opakuj omej nádobí dokuď, aby bylo čistý. Nemůžu říct omej 3x nádobí. Nevíme, kolikrát to má omýt.“* **Žák 2** – *„Dokud třeba nebude smetí pryč. Protože jsme nevěděli kolik tam je třeba smetí.“*

4. ročník: Žák 1 – *„Protože jsme nevěděli, kolikrát to máme udělat. Třeba kolikrát máme posbírat kostičku.“* **Žák 2** – *„Protože třeba někdy nevíme kolik tam toho je, kolikrát to máme udělat.“* **Žák 3** – *„Když jsem třeba nevěděl, kolik tam něčeho je, když jsem měl třeba něco*

brát, tak jsem nevěděl, kolik tam toho je. Sem nemoh dát Opakuj, nevěděl jsem počet.“ **Žák 4** – *„Protože jsme nevěděli, kde je cíl.“*

5. ročník: Žák 1 – *„Protože jsme nevěděli, kde je start a kde cíl.“* **Žák 2** – *„Protože to bylo jednodušší, protože jsme tam dali třeba krok dopředu, dokud tam není zeď. Robot by nevěděl, kolikrát má udělat krok dopředu.“*

Otázka 6 se týkala ověření toho, jestli žáci správně pochopili smysl použití opakovacího příkazu „*Opakuj – Dokud*“. Při programování je důležité, aby se žáci oprostili od vizuální představy a pochopili, že instrukce v podobě sestaveného algoritmu za ně bude provádět stroj, který neví, jak je například dráha dlouhá nebo kolik se na dráze nachází kostiček, které má za úkol sebrat. Metodicky to lze vyřešit například aktivitou, při které má dítě šátkem zavázané oči. Všichni žáci na tuto otázku odpověděli uspokojivě. U mladších žáků musel být kladen důraz na to, aby se snažili vysvětlit, proč museli použít tento příkaz, jelikož se snažili uvádět spíše příklady použití příkazu „*Opakuj – Dokud*“.

Otázka 7: Jak robot pozná, kolikrát má opakovat základní příkaz (např. krok) při použití opakovacího příkazu *„Opakuj – Dokud“*?

3. ročník: Žák 1 – *„Třeba dokuď nebude čistý. Běž dopředu, dokud tam bude volno. Opakuj seber kostičku, dokud nebude prázdno, dokud tam nic nezbyde, smetí = 0.“* **Žák 2** – *„Dokud nebude smetí = 0.“*

4. ročník: Žák 1 – *„Třeba dokud je smetí rovno 0. Dokud se tohle nesplní.“ (*pozn. autora – podmínka) **Žák 2** – *„Dovnitř příkazu se dá to, co se má opakovat a sem dokdy.“* **Žák 3** – *„Dáme tam opakovat, a co máme opakovat, a pak dáme dokud. Třeba dokud tam je díra, tak ji máme vyplnit.“* **Žák 2** – *„Opakuj krok dopředu, dokud tam nebude kostička. Opakuje, dokud něco.“*

5. ročník: Žák 1 – *„Dokud třeba nebude nějaká zeď.“* **Žák 2** – *„Dokud tam třeba není ta hromádka.“*

U opakovacího příkazu „*Opakuj – Dokud*“ je velmi důležité, aby žáci dokázali objevit a nastavit omezení, které cyklus ukončí (*Dokud*). Očekávalo se, že žáci budou odpovídat v tomto duchu. Otázka zřejmě nebyla pro žáky dosti srozumitelná a žáci v odpovědích na tuto otázku uváděli příklady použití, což patřilo spíše do následující otázky.

Otázka 8: Uveďte nějaké podmínky (*Dokud*), které opakování příkazu *Opakuj – Dokud* ukončí.

3. ročník: Žák 1 – *„Opakuj sněz, jez polévku, dokuď nebude prázdná, dokuď polévka nebude prázdná, dokuď miska nebude prázdná. Zalejvej kytku, dokuď to nebude plný. Mistička, váza, květináč.“* **Žák 2** – *„Třeba když je slepá ulice, tak opakuj otoč se, dokud nebude možná cesta, že se otočí na druhou stranu.“*

4. ročník: Žák 1 – *„Opakuj seber, dokud tam nějaké smetí je. Nebo třeba že bysme měli jít jeden krok dopředu, dokud nedojdeme ke zdi. Lij vodu do květináče, dokud nebude plný.“* **Žák 2** – *„No že třeba když mám rovinu, tak nevím třeba kolik je tam polí, kolik musím udělat kroků dopředu, tak zase musím dát opakuj dokud nebudu třeba u zdi nebo u zatáčky. Opakuj nabrat polévku lžící, dokud nebude snědená, nebo dokud tam žádná nebude nebo dokud v talíři nezbyde polévka.“* **Žák 3** – *„Když tam dám to opakuj dokud, tak vlastně než tam bude někde, on tam dojede, i když neví kolik tam toho je, těch kroků. Nebo když tam mám třeba nějaký počet kostiček, jako třeba 5 – 10, a nevím teda kolik jich je, tak nedám opakuj, ale opakuj dokud tam žádná nebude.“* **Žák 4** – *„Sbírej smetí, dokud tam žádný nebude. Opakuj nabrat polévku na lžíci, dokud nebude talíř prázdnej. Nalít vodu do květináče, dokud kytka nevysaje všechnu vodu (smích), dokud nebude květináč plnej.“*

5. ročník: Žák 1 – *„Dokud třeba nebude překážka. Opakuj naber polévku na lžíci, otevři pusu, dej lžíci do pusy, dokud nebude talíř plný ……. ne teda prázdný. Dokud nebude nastartovaná motorka. Dokud nebudou vypočítaný všechny příkazy.“* **Žák 2** – *„Opakuj krok dopředu, dokud není zeď. Robot půjde furt dopředu, dokud nenarazí.“*

Otázka 8 měla přispět ke zjištění, jestli žáci dokáží vymyslet nebo si vzpomenout na různá omezení, která by ukončila cyklus (*Opakuj – Dokud*). Některým žákům byly také kladeny doplňující otázky, které se týkaly nastavení omezení u běžných činností v reálném životě, jako například jezení polévky (dokud nebude talíř prázdný) nebo zalévání květiny (dokud nebude květináč plný).

Otázka 9: Dokážete říci, kdy jste museli použít příkaz *Pokud*?

3. ročník: Žák 1 – *„Pokud kytka bude suchá, tak ji zalij. Pokud tam je smetí, tak ho seber.“* **Žák 2** – *„To si nevzpomenu.“* (Po nápovědě) *„Pokud je smetí > 0, tak ho seber.“*

4. ročník: Žák 1 – *„Třeba jakoby, když bysme chtěli někam jít, tak pokud je tam cesta, tak udělej jeden krok dopředu, a kdyby tam ta cesta nebyla, tak nic neudělá.“* **Žák 2** – *„Že třeba pokud tam je místo, udělej krok dopředu. Když tam to místo není, tak ho neudělá.“* **Žák 3** – *„Pokud bude na dráze nějaké smetí nebo kostička, tak seber. A pokud tam nebude, tak ho nebude sbírat.“* **Žák 4** – *„Pokud tam je třeba zatáčka, tak tam zatočí.“*

5. ročník: Žák 1 – *„Pokud je překážka, kostička, tak ji seber.“* **Žák 2** – *„Pokud narazíš na kostičku, tak ji seber.“*

Otázka 9 se týkala ověření toho, jestli žáci správně pochopili smysl použití příkazu *Pokud.* U příkazu *Pokud* program nejdříve zjišťuje, jestli je splněná podmínka příkazu, a pokud ano, tak provede příkaz, který se nachází v těle příkazu. Pokud podmínka splněná není, neprovede se žádný příkaz, eventuálně program pokračuje dalším příkazem, který se nachází za příkazem *Pokud*. Kromě jednoho žáka, který si na použití příkazu Pokud vzpomněl až po drobné nápovědě, si ostatní testovaní žáci správně vybavovali a uváděli příklady použití tohoto příkazu.

Otázka 10: Co udělá vozidlo (robot), když není splněna podmínka u příkazu *Pokud*?

3. ročník: Žák 1 – *„Tak nic neudělá.“* **Žák 2** – *„Nic.“*

4. ročník: Žák 1 – *„Neudělá nic.“* **Žák 2** – *„Neprovede nic.“* **Žák 3** – *„Ten příkaz provede, jen když je to splněný.“* **Žák 4** – *„Tak to neprovede.“*

5. ročník: Žák 1 – *„Tak nebude dělat nic a nevadí to.“* **Žák 2** – *„Tak se nic neprovede.“*

Otázka 10 měla ověřit, jestli žáci správně pochopili, jak se bude chovat program, pokud podmínka příkazu *Pokud* splněna nebude. Pokud podmínka splněna není, neprovede se příkaz ve struktuře příkazu „*Pokud“* a program bude pokračovat. Na tuto otázku všichni žáci odpovídali správně a bez váhání.

Otázka 11: Vysvětlete, jaký je rozdíl při použití příkazu *Pokud* a příkazu *Opakuj - Dokud*.

3. ročník: Žák 1 – *„Že tady (Pokud) ta podmínka se splní jako první, a když se splní ta podmínka, tak se udělá tady to. A když se nesplní, tak se neudělá nic. A tady (Opakuj – Dokud) se nejdřív provede něco a pak teprve tohle (podmínka příkazu). Tadyten (Opakuj –*

Dokud) se bude opakovat a tady (Pokud) to proběhne jenom jednou.“ **Žák 2 –** *„Že tady (Opakuj – Dokud) se opakuje a tady (Pokud) se jenom dělá, proběhne jenom jednou. Tady (Opakuj – Dokud) provede nejdřív ten příkaz, a pak tu podmínku. Tady (Pokud) nejdřív podmínku, a pak ten příkaz. U opakuj dokud se provede aspoň jednou.“*

4. ročník: Žák 1 - *„Že Opakuj – Dokud je opakovací a Pokud není. Takže tady se to bude něco opakovat, dokud tam nebude třeba cíl. Tady u Pokud, když tam třeba bude kostička, tak to sebere a když tam nebude, tak to neudělá. U Opakuj – Dokud tenhle příkaz udělá aspoň jednou, protože se opakuje dokud, u Pokud ne, když tam ta kostička nebude, tak to neudělá.“* **Žák 2 –** *„Tadyhle (Pokud) třeba nejdřív se ten robot koukne na tu podmínku, a pak až něco udělá. A tady (Opakuj – Dokud) nejdřív něco udělá, a pak až jako sleduje tu podmínku. Tohle (Pokud) proběhne jenom jednou a tohle (Opakuj – Dokud) se opakuje, třeba dokud je na silnici díra, tak ji vyplň.“* **Žák 3 -** *„To Opakuj je, že to bude dělat víckrát, do nějaký určený doby, dokud nebude mít nějaký příkaz, kterej mu dá vědět, že už má přestat. A to pokud, v tom je rozdíl že třeba, že pokud bude nějaká podmínka, která mu dá nějaký povel. Tady (Opakuj – Dokud) to má dělat víckrát a tady jenom jednou. Tady (Pokud) je nejdřív ta podmínka a tady (Opakuj – Dokud) je až na konci. Udělá se nejdřív něco, a pak až bude vědět dokdy.“* **Žák 4 –***„Pokud jako ……. Pokud tam bude nějaká kostka, tak třeba ji sebrat. A když tam není tak nesebrat. Opakuj je, že to třeba udělá, dokud je třeba cíl. Dokud není třeba cíl, tak musí třeba krok dopředu. Příkaz Pokud neopakuje.“*

5. ročník: Žák 1 – *„Opakuj dokud, to je, že nejdřív zadáme úkol a pak se to musí nějak dodělat, dokud to nebude hotový, a pokud je ta podmínka na začátku.“* **Žák 2 –** *„Že tady (Opakuj – Dokud) se to opakuje několikrát, a tady (Pokud) se to opakuje jenom jednou. Že tady (Opakuj – Dokud) bude třeba opakovat krok dopředu dokud, třeba do cíle. Tady (Pokud) se počítač podívá nejdřív na tu podmínku, když je splněná, bude dělat, a když není splněna, tak se nic nestane.“*

Otázka 11 měla ověřit, jak žáci vnímají průběh programu při použití cyklu *Opakuj – dokud* a příkazu s testovací podmínkou *Pokud.* U příkazu *Pokud* se nejprve vyhodnotí podmínka, a pokud je splněna, tak se provede příkaz v těle tohoto příkazu. Základní příkaz v těle příkazu *Pokud* se pak nemusí provést ani jednou. Naopak u cyklu *Opakuj – Dokud* se nejprve provede základní příkaz v těle cyklu a teprve potom se testuje podmínka příkazu (*Dokud*).

U cyklu *Opakuj – Dokud* se pak základní příkazy v těle cyklu provedou vždy alespoň jednou.[41] Žáci si také měli uvědomit, že příkaz *Opakuj – Dokud* je na rozdíl od příkazu *Pokud*, opakovací příkaz.

Mladší žáci, zejména pak žáci 3. ročníku, se velmi špatně vyjadřovali. Sice u nich bylo patrné, že si rozdíl v průběhu programu při použití těchto příkazů uvědomují, ale jejich malá slovní zásoba a vyjadřovací schopnosti jim nedovolily se nějakým smysluplným způsobem vyjádřit. Formulace otázky 11 nebyla nejspíše pro vybrané žáky příliš srozumitelná, jelikož řada z nich nepochopila, že je po nich požadováno porovnání průběhu programu při použití těchto dvou příkazů. Musely jim být položeny doplňující otázky, v nichž se kladl důraz na to, aby se žáci zaměřili právě na to, které kroky program provede při použití těchto dvou příkazů a samozřejmě za jakých podmínek. Žáci, kteří měli s odpovědí obtíže, mohli pro vysvětlení používat kartičky s příkazy *Opakuj – Dokud* a *Pokud*.

Otázka 12: V jakém případě se provede u příkazu *Pokud – Jinak* jeden základní příkaz a kdy druhý základní příkaz? (Např. *Pokud je cesta vpřed – krok dopředu, jinak – otočit vlevo*)

3. ročník: Žák 1 – *„Když bude podmínka splněná, tak se udělá tenhle první, a když nebude podmínka splněná, tak se udělá tenhle druhej.“* „ ***Žák 2*** – *„(Pokud – Jinak) Když tohle bude splněno, tak se tohle udělá, a když nebude splněná, tak jinak něco jinýho.“*

4. ročník – Žák 1 – *„Když by tam byla zeď, tak se tenhle první příkaz neprovede a provede se tenhle a on se otočí. Když bude splněná podmínka, tak se provede tento příkaz, když nebude, tak tadyten.“* **Žák 2** – *„No když se ta podmínka splní, provede se tohle, a když se nesplní, tak se provede tohle.“* **Žák 3** – *„Když bude splněná (podmínka), tak se provede tenhle příkaz, tak se neprovede tenhle ani tenhle, teda tak provede se tenhle druhý.“* **Žák 4** – *„Pokud tam je zas ta kostička, tak ať ji sebere, jinak když tam není, tak ať ji nesebere. Pokud je cesta vpřed, tak má jít dopředu, se otočit vlevo.“*

5. ročník - Žák 1 – *„Když třeba je tady, pokud je cesta dopředu, tak udělá krok dopředu a když není, tak třeba zatočí doprava.“* **Žák 2** – *„Že když tady bude, pokud je třeba venku*

[41] FABIAN, D. Podmínky a cykly. [online] 2013. Dostupné z: http://kmlinux.fjfi.cvut.cz/~fabiadav/cecko/poznamky-k-jazyku-c/podminky-a-cykly

zima, tak když je podmínka splněná, tak se udělá ten první příkaz, a když není splněná, tak se udělá ten druhej příkaz.“

Žáci kvůli snazšímu pochopení nejdříve používali samotný příkaz *Pokud* (bez větve *Jinak)*. Otázka 12 měla ověřit, jestli žáci pochopili průběh programu u podmiňovacího příkazu *Pokud* s přidanou větví *Jinak*. S odpovědí na tuto otázku žáci problémy víceméně neměli.

Otázka 13: Mohou se u příkazu *Pokud – Jinak* provést oba dva základní příkazy?

3. ročník - Žák 1 – *„Ne. Protože tady je ta podmínka. Když je splněná, tak se může udělat jenom jeden, a když není splněná, tak se může udělat ten druhej.“* **Žák 2** – *„Ne. Buď se provede jeden, nebo druhej.““*

4. ročník – **Žák 1** – *„Ne.“* **Žák 2** – *„Ne“* **Žák 3** – *„Ehm….. Ne?“* **Žák 4** – *„Ne.“*

5. ročník – **Žák 1** – *„Ehm… Jo?“* V jakém případě? *„Když třeba ….. nevim.“* **Žák 2** – „… ne.“

Otázka 13 měla stejně jako předchozí otázka ověřit, jestli žáci správně pochopili průběh programu u podmiňovacího příkazu *Pokud* s přidanou větví *Jinak*. Jedna žákyně pátého ročníku měla s touto otázkou problémy, i když na předchozí otázku odpověděla dobře. Po té, co byla této žákyni znovu položena předchozí otázka a znovu ji správně zodpověděla, odpověděla na tuto znovu položenou otázku již správně.

Otázka 14: Vysvětlete, jaký je rozdíl při programování mezi použitím příkazu *Pokud* a *Pokud - Jinak*.

3. ročník - Žák 1 – *„Tady máš jenom pokud a tady máš pokud, jinak. Tady u toho pokud, když nebude podmínka splněná, tak se neudělá nic, jenomže tady, když nebude podmínka splněná, tak se udělá tahle (větev Jinak).“* **Žák 2** – *„ Tady u pokud, když nebude podmínka splněná, tak se neudělá nic, a tady když nebude splněná, tak se něco udělá.“*

4. ročník – **Žák 1** – *„Tady u Pokud, když tam ta podmínka nebude splněná, tak neudělá nic jinýho, a tady u Pokud – Jinak udělá ten druhej příkaz.“* **Žák 2** – *„U toho pokud je jenom jako když není splněna, nic neudělá. Ale když tady (Pokud – Jinak) není splněná, tak udělá něco jinýho.“* **Žák 3** – *„Stejnýho je, pokud je ta podmínka splněná, tak se to vykoná, nějakej příkaz, kterej bude tady. Když ta podmínka nebude splněná, tak tady (Pokud) se neprovede*

nic, a tady (Pokud – Jinak) se může provést třeba něco, třeba udělej krok dopředu, nebo nějakej jinej příkaz.“ **Žák 4 –** *„Když tady (Pokud) není podmínka splněná, tak nedělá nic, a tady taky nic.“* Jelikož tento žák neodpověděl správně, byla mu otázka položena znovu, ale měl k dispozici sestavený algoritmus z „unplugged“

úlohy č. 8 (***Pokud smetí>0 – seber1, jinak – krok dopředu***) pomocí materiálních kartiček s příkazy. *„Když tam je nějaká kostička, tak sebere jednu, a když tam není, tak jde krok.“*

5. ročník – Žák 1 – *„Tady (Pokud – Jinak) může ještě to jinak. Tady (Pokud), když je podmínka splněná, tak to udělá, ten krok, a když není, tak to neudělá, nic neudělá. Tady (Pokud – Jinak), když je podmínka splněná, tak to udělá, ten příkaz, a když není splněná, tak udělá to jinak.“* **Žák 2 –** *„Tady (Pokud), když je splněná ta podmínka, tak se udělá ten příkaz, a když není splněná, tak se nic nestane. A tady (Pokud – Jinak), když je splněná ta podmínka, tak se udělá ten příkaz, ten první, a když není splněná, tak se udělá ten druhej přikaz.“*

Otázka 14 měla ověřit, jak žáci rozumějí průběhu programu při použití příkazu *Pokud* a příkazu *Pokud – Jinak.* U příkazu *Pokud* i *Pokud – Jinak* se nejprve testuje podmínka příkazu a pokud je splněna, provede se „první“ příkaz, a pokud podmínka splněna není, tak se tento příkaz neprovede. U příkazu Pokud – Jinak se na rozdíl od příkazu Pokud provede příkaz ve větvi Jinak, pokud podmínka splněna není, až po té program pokračuje dalšími následujícími příkazy. U příkazu Pokud program pokračuje dalšími následujícími příkazy ihned, jakmile podmínka příkazu splněna není.

Otázka 15: Když se příkaz *Pokud* nebo *Pokud - Jinak* neopakuje, je možné ho v programu nějakým způsobem opakovat? Jestli ano, tak jakým způsobem?

3. ročník: Žák 1 – *„Můžou, když je vložíme do opakovacích příkazů.“* **Žák 2 –** *„Jo, tady (kolem) bysme mohli dát větev opakuj.“*

4. ročník: Žák 1 – *„Asi bych tam dala nějakej ten příkaz opakuj dokud nebo opakuj třeba 5x.“* **Žák 2 –** *„Třeba opakuj dokud nebo opakuj třeba 2x.“* **Žák 3 –** *„Dát to na to opakuj. Třeba opakuj dokud nebo samotný opakuj.“* **Žák 4 –** *„Zavřít to do opakuj, opakuj kolikrát a opakuj dokud.“*

5. ročník: Žák 1 – *„Když bysme do toho dali, okolo toho dali to opakuj. Kdyby sme to dali do toho opakuj.“* **Žák 2** – *„ …… ne? Asi jo.“* Jak? *„Nooo, že bude třeba opakuj dokud nebo opakuj několikrát.“*

Otázka 15 měla ověřit, jestli žáci pochopili, že mohou do jednotlivých cyklů vnořit celé bloky jiných příkazů, a tím pádem mohou *Pokud - Jinak* také opakovat.

Z analýzy odpovědí žáků na jednotlivé otázky obecně vyplývá, že žáci po absolvování navržené výuky, která se opírá o „unplugged“ aktivity a na ně navazující aktivity v programovacím prostředí Code.org, správně pochopili smysl, funkční princip a způsob použití vybraných algoritmických konceptů *Opakuj Xkrát, Opakuj – Dokud, Pokud* a *Pokud – Jinak*. Rozhovor s některými žáky, zejména se žáky 3. ročníku, byl náročný, jelikož se hůře soustředili a mnohdy hůře chápali smysl položených otázek. Nejspíše kvůli slabé slovní zásobě a nedostatečným vyjadřovacím schopnostem jim také dělalo problémy se ve svých odpovědích srozumitelně vyjádřit a mnohdy bylo těžké určit, co svou odpovědí myslí. Vyjadřování starších žáků (z vyšších ročníků) bylo z jazykového hlediska bohatší a přesnější. Žáci mnohdy nedokázali odpovídat obecně a při svých odpovědích používali konkrétní příklady, které se většinou týkaly úloh, které řešili při výuce. Jejich odpovědi pak byly přesnější, jasnější a konkrétnější, pokud odpovídali na otázky týkající se jednodušších algoritmických konceptů, jako jsou posloupnost základních příkazů a cykly s pevným počtem opakování (*Opakuj Xkrát*). Žáci 4. a 5. ročníku pak měli obecně s pochopením smyslu otázek a vyjádřením ve svých odpovědích menší problémy než žáci 3. ročníku. Zřejmě tento fakt souvisí s věkem a rozvojem jejich mentálních dovedností, čtenářskou gramotností, s úrovní jazykového vyjadřování, s rozvojem abstraktního myšlení. Jelikož pro řízené rozhovory byli žáci vybíráni náhodně, byla mezi dotazovanými i žákyně s individuálním vzdělávacím programem. S touto žákyní rozhovor neproběhl úspěšně. I když věděla, že nejde o zkoušení na známky, ale pouze o rozhovor s učitelem, byla značně rozrušená a naprostou většinu položených otázek nepochopila a nedokázala na ně odpovědět nebo odpovídala špatně. Při plnění úloh v rámci „unplugged“ aktivit však s ostatními žáky komunikovala bez problémů a také z analýzy řešení úloh v Code.org vyplynulo, že ani s úlohami ve virtuálním programovacím prostředí neměla výraznější problémy. Žákyni

nemusel vyhovovat způsob dotazování v řízeném rozhovoru s učitelem, nebo jí úroveň jejího abstraktního myšlení nedovolovala obecněji mluvit o smyslu a principu použití vybraných příkazů. U této žákyně se jednalo o odpovědi na otázky související s příkazy *Pokud* a *Pokud – Jinak.*

6.4 Výsledky a analýza tvorby vlastních bloků příkazů

Z analýzy nashromážděných dat pomocí průběžného zapisování poznatků z hodin a fotodokumentace výsledků práce žáků při tvorbě a sestavování vlastních bloků příkazů vyplynulo, že žáci správně porozuměli a pochopili smysl a funkční princip použití vybraných algoritmických konceptů a dokázali víceméně bez problémů přepsat větu z mateřského jazyka do jazyka „srozumitelného pro stroje“. Žákům činila problém správná formulace omezené instrukční sady základních příkazů, které by mohl robot (stroj) provádět. Jednalo se o situace, kdy žáci spíše popisovali události, které nastanou při splnění nebo eventuálně nesplnění testovací podmínky příkazu (Například: „Pokud přijdeš včas, dostaneš odměnu, jinak dostaneš výprask.“). Někteří žáci měli také problémy s vyjádřením hodnoty proměnné v testovací podmínce (Například: smetí > 0). Tyto dva zmiňované jevy nebyly dále důkladněji zkoumány. Analýza těchto jevů by byla nad rámec zadání diplomové práce. Pro ověření funkčnosti navržených metodických postupů bylo důležité, že žáci správně pochopili, jak bude program probíhat při použití příkazů s testovacími podmínkami, dokázali nastavit omezení, které ukončí cykly *Opakuj Xkrát* a *Opakuj - Dokud*, nebo nastavit testovací podmínku u příkazů *Pokud* a *Pokud – Jinak* a definovat, co se stane, když testovací podmínka bude nebo nebude splněná. Žáci jedné skupiny museli být usměrňováni, jelikož vymýšleli věty s nevhodným, až nevkusným, obsahem.

Obr. 37 - Ukázka tvorby vlastních bloků příkazů

6.5 Návrh na úpravy

Na základě analýzy nashromážděných dat byly zjištěny obtíže, s nimiž se žáci při výuce potýkali, a na základě toho byly navrhnuty změny v realizovaných „unplugged“ aktivitách, aktivitách ve virtuálním programovacím prostředí Code.org a v organizaci výuky.

Navržené úlohy s Ozobotem byly pro žáky sice velmi motivující, avšak časově náročnější než se předpokládalo, a bylo obtížné je splnit během jedné vyučovací hodiny. Pro plynulejší průběh výuky by bylo vhodné buď snížit počet úkolů nebo navýšit aktivitám s Ozobotem časovou dotaci na dvě vyučovací hodiny.

Ukázalo se, že žáci, kteří měli zadané úlohy vyřešené mnohem rychleji než ostatní, se začali rychle nudit a svými činnostmi dokázali narušovat plynulý průběh výuky. Při „unplugged“ aktivitách bylo rychlejším skupinám upraveno zadání úloh v podobě doplňkových úkolů, ve kterých měl pomyslný sběrač odpadků sebrat větší počet kostiček z lega představující smetí na různých místech vyznačené dráhy. V Code.org bylo využito možnosti zpřístupnění rozšiřujících a složitějších úloh pro žáky, kteří již požadovanou lekci dokončili. Díky tomu rychlejší skupiny a žáci nedostávali prostor pro to, aby se nudili a rušili ostatní pomalejší skupiny nebo jednotlivé žáky.

Příprava materiálních didaktických pomůcek v podobě kartiček s příkazy byla relativně časově náročná kvůli velkému počtu různých velikostí kartiček, zejména kartiček představujících příkazy s testovacími podmínkami. V průběhu pedagogického experimentu se ukázalo, že některé velikosti kartiček s příkazy nebyly použity nebo naopak některé velikosti těchto kartiček chyběly. Tyto kartičky by bylo vhodnější navrhnout tak, aby se mohla jejich velikost variabilně měnit a vkládat do nich libovolný počet jiných příkazů. Původní návrh počítal s tím, že budou do pedagogického experimentu zařazeny také „unplugged“ aktivity s kartičkami s příkazy zaměřené na výuku funkcí, aby navržená výuka odpovídala celému rozsahu připravované aktualizace RVP v oblasti algoritmizace a programování. Po pilotních lekcích bylo zřejmé, že postupem času jsou již „unplugged“ aktivity pro žáky nezajímavé a že žáci budou upřednostňovat aktivity v Code.org.

Aktivity, při kterých žáci přepisovali věty z mateřského jazyka do jazyka „srozumitelného pro stroje“ s použitím kartiček s příkazy a vytvářeli tak vlastní bloky příkazů, ve kterých používali příkazy s testovacími podmínkami, nedělaly žákům větší obtíže. V každém

případě se osvědčilo, aby žáci pracovali s ukázkami z běžného života. Z hlediska ověřování správného pochopení použití příkazů s testovacími podmínkami při sestavování algoritmů bylo zařazení těchto aktivit do pedagogického experimentu vhodné, avšak pro rozvoj programátorských dovedností by bylo vhodné věnovat těmto aktivitám menší časový prostor (cca 15 minut) a více se zaměřit na řešení úloh ve virtuálním programovacím prostředí.

Navržená výuka probíhala ve tříhodinových blocích. Po dokončení a analýze výsledků pedagogického experimentu se ukázalo, že lze výuku založenou na navržených metodických postupech bez problémů aplikovat i s časovou dotací jedna hodina týdně, a že takto rozdělená výuka nemá na výsledky práce žáků víceméně žádný vliv. Výhoda blokově orientované výuky spočívá zejména v tom, že při navržených „unplugged“ aktivitách je možné dokončit nedořešené úlohy bez ohledu na časové omezení vyučovací hodiny nebo je možné „rozestavěný“ algoritmus nechat rozpracovaný na stole přes přestávku, což při výuce, která probíhá s časovou dotací jedna hodina týdně, není možné. Totéž podobně platí i pro aktivity ve virtuálním programovacím prostředí.

7 Závěr

Ve výzkumné části diplomové práci byl aplikován kvalitativní výzkum s využitím metody akčního výzkumu. Pedagogický experiment byl realizován v malé vesnické škole s malým počtem žáků, tudíž nelze zjištění získaná z pedagogického experimentu zobecnit. Lze se ze zjištění poučit a inspirovat.

Ukázalo se, že žáci velmi rychle ztrácejí motivaci a chuť k práci, pokud jim něco nejde. To samé platí i o sestavování algoritmů a programování. Pokud se při výuce programování nebude přistupovat k žákům citlivě, spoustu žáků tím dokážeme odradit. Vytvořením „mostu“ mezi reálným světem a virtuálním světem programování pomocí „unplugged“ aktivit docílíme toho, že následná práce žáků ve virtuálním prostředí pro ně nebude obtížná. Cílem diplomové práce bylo navrhnout takové metodické postupy, které by byly srozumitelné pro žáky 1. stupně ZŠ, které by jim pomohly pochopit smysl a princip použití vybraných algoritmických konceptů při sestavování algoritmů a které by dokázaly propojit „unplugged“ aktivity s aktivitami ve virtuálním programovacím prostředí tak, aby si žáci osvojili určitý soubor algoritmických dovedností.

Vyhodnocení výsledků práce a ověření funkčnosti navržených metodických postupů bylo provedeno podrobnou analýzou nashromážděných dat prostřednictvím průběžného zapisování poznatků a videozáznamů zachycujících práci žáků a problémy při průběhu řešení jednotlivých úloh v rámci navržených „unplugged“ aktivit, analýzou pokroku žáků ve vybraném virtuálním blokově orientovaném programovacím prostředí, analýzou audiozáznamů odpovědí žáků na otázky při řízeném rozhovoru a analýzou fotodokumentace výsledků ověřování dovedností žáků sestavovat vlastní bloky příkazů, ve kterých žáci přepisovali věty z mateřského jazyka do řeči „srozumitelné pro stroje“, do programovacího jazyka. Jelikož rodiče žáků podepisovali informovaný souhlas (Příloha 6), kde bylo uvedeno, že veškerá získaná data budou anonymní, nebudou zveřejněna a budou použita pouze k vyhodnocení výzkumného šetření, jsou nashromážděná data v podobě audio a video záznamů k nahlédnutí u autora diplomové práce.

Po analýze nashromážděných dat lze usuzovat, že použité metodické postupy, které se opírají o navržené „unplugged“ aktivity a na ně navazující navržené aktivity ve virtuálním

programovacím prostředí Code.org, splnily svůj účel, jsou funkční a byly pro žáky zábavné a motivující. Žáci se na hodiny výuky algoritmizace těšili, dávali to vyučujícímu najevo a někteří žáci složitější doplňkové úlohy v Code.org řešili dobrovolně doma a následně se chlubili sestavenými algoritmy. Žáci zejména ze 4. a 5. ročníků správně porozuměli smyslu použití a funkčnímu principu příkazů s testovacími podmínkami při sestavování algoritmů, dokázali si uvědomovat, které události nastanou v jednotlivých krocích programu, čemuž velkou měrou přispěla navržená kombinace skupinových činností v rámci „unplugged" aktivit (sestavování algoritmů s využitím kartiček s příkazy a následné manuální přehrávání sestavených algoritmů žáky) se samostatnou činností v Code.org. Zejména žáci 4. a 5. ročníků dokázali bez větších problémů znalosti a dovednosti získané v rámci „unplugged" aktivit uplatňovat v Code.org, čímž docházelo k požadovanému propojení mezi „unplugged" aktivitami a aktivitami ve virtuálním programovacím prostředí. Náklady na pořízení materiálních pomůcek v podobě kartiček s příkazy jsou minimální a každý učitel si je dokáže sám jednoduše připravit. Kartičky s příkazy byly vytištěny na silnější papír, zafoliovány a vystřiženy. Poničené kartičky s příkazy lze snadno a rychle nahradit novými, avšak po použití při „unplugged" aktivitách byly poničeny minimálně, a lze konstatovat, že takto připravené kartičky jsou i relativně odolné.

Při plnění úkolů navržených ve virtuálním prostředí Code.org žáci pracovali každý na svém počítači v modelu 1:1. Takováto výuka samozřejmě přispívá k rozvoji individuálních algoritmických dovedností jednotlivých žáků, na druhou stranu je pro učitele poměrně složité se věnovat více žákům najednou, pokud se u nich objeví nějaké problémy a mají otázky spojené s řešením dané úlohy. Práce s kartičkami tento problém dokáže relativně jednoduše odstranit. Učitel se může se žáky, kteří mají nějaký problém při řešení úlohy ve virtuálním prostředí, vrátit zpět k „unplugged" aktivitám s kartičkami. Učitel jednoduše zadá podobný úkol tomu, ve kterém mají žáci problém, a společně se pokouší tento problém vyřešit. Žáci následně opět manuálně přehrávají sestavený algoritmus, a tím dochází ke snazšímu a víceméně abstraktnímu (obecnému) pochopení problému a jeho transferu do virtuálního prostředí. Tímto způsobem žáci přicházejí na to, kde udělali chybu a na správné řešení sami, a tím dochází k rozvoji algoritmického myšlení a programátorských dovedností.

Z poznatků z výuky lze usuzovat, že „slabší“ a pomalejší žáci byli ve skupinách při „unplugged“ aktivitách nuceni pracovat v rychlejším tempu. U některých „slabších“ žáků se zpočátku objevovaly tendence k neakceptování rychlejšího tempa, naopak „silnější“ žáci nechtěli pouštět „slabší“ žáky k „unplugged“ aktivitám. Tito žáci se pak dostávali pouze do role diváků a začínali být znechuceni. Tomuto jevu bylo důležité hned na začátku ze strany učitele zabránit a žákům vysvětlit, že pracují ve skupině a úspěch skupiny závisí na všech členech skupiny. Jelikož žáci neměli mnoho dosavadních zkušeností se skupinovou prací, byl tento pozorovaný jev očekávatelný a pochopitelný. Postupem času již k tomuto nedocházelo a žáci ve skupinách spolupracovali, „šikovnější“ žáci vysvětlovali nepochopené věci „slabším“ a docházelo tak k požadovanému vrstevnickému učení.

Ukázalo se také, že při řešení úloh v Code.org byli žáci většinou hluboce ponořeni do problému, byli maximálně soustředěni a koncentrováni na vyřešení úkolu a nebyli schopni ani vnímat informace a instrukce učitele, které se týkaly často objevovaných chyb. V těchto situacích se ukázalo dobré přerušit práci žáků a zařadit krátkou „unplugged“ aktivitu, která se zaměřovala právě na tyto opakované chyby. Při sestavování algoritmů v Code.org žáci neměli obtíže se správným použitím příkazů s testovací podmínkou. Někteří žáci měli ze začátku při sestavování algoritmů drobné obtíže s pravolevou orientací při řízení pohybu objektu na počítači. Po několika lekcích si již tito žáci správně uvědomovali směrové nastavení virtuální postavy a dokázali z hlediska pravolevé orientace sestavovat algoritmy bez chyb. U složitějších úloh, dělalo zejména mladším žákům 3. ročníku obtíže najít v sestaveném algoritmu větší části algoritmu, které by mohli nadále opakovat a zkrátit si tím zápis algoritmu, což po nich bylo v Code.org požadováno prostřednictvím předepsaného počtu použitých bloků příkazu. Algoritmus měli většinou funkční, ale použili větší počet bloků příkazů, než po nich bylo požadováno.

Vzhledem k tomu, že se výrazněji projevila odlišnost v počtu a úspěšnosti vyřešených úloh v Code org mezi žáky 3. ročníku a žáky vyšších ročníků, bylo by vhodné, aby žáci 3. ročníku řešili jednodušší úlohy týkající se pouze posloupnosti základních příkazů, eventuálně úlohy s použitím cyklu *Opakuj Xkrát.* Úlohy zaměřené na sestavování algoritmů s využitím složitějších příkazů s testovacími podmínkami (*Opakuj – Dokud, Pokud, Pokud – Jinak*) by pak bylo vhodné zařadit do výuky až ve 4. nebo 5. ročníků. Nejjednodušší úlohy v Code.org

v kurzu 1[42], zaměřené na posloupnost základních příkazů, by pak byly vhodné i pro nejmladší žáky základní školy. Pro rozvoj algoritmického myšlení žáků 1. stupně ZŠ je v současné době připravováno prostředí EMIL, na jehož vývoji se podílejí prof. Ivan Kalaš a dr. Andrej Blaho z Univerzity Komenského v Bratislavě.

Závěrem lze konstatovat, že použité metodické postupy opírající se o navržené „unplugged" aktivity a na ně navazující aktivity ve virtuálním programovacím prostředí Code.org splnily svůj účel, lze je doporučit pro výuku algoritmizace a základů programování na 1. stupni ZŠ a dosažené výstupy žáků po absolvování navrženého pedagogického experimentu odpovídají požadovaným výstupům oblasti algoritmizace předpokládané aktualizace RVP ZV. V úvodních fázích výuky programování lze doporučit vícehodinové bloky, aby i „slabší" žáci měli prostor pro to, aby dokázali rozpracované úlohy (i po přestávce) dokončit a mohli mít ze své práce radost. Bezplatné virtuální blokově orientované programovací prostředí Code.org lze také doporučit pro učitele, kteří nemají s výukou programování žádné zkušenosti a začínají základy programování učit.

[42] https://studio.code.org/s/course1

8 Seznam použitých informačních zdrojů

ACKERMANN, E. Piaget's Constructivism, Papert's Constructionism: What's the difference? In: Future of learning group publication, 2001, vol. 4, no. 3. Dostupné z: http://learning.media.mit.edu/content/publications/EA.Piaget%20_%20Papert.pdf

BASL, J., BOUDOVÁ, S., ŘEZÁČOVÁ, L. Národní zpráva šetření ICILS 2013. Počítačová a informační gramotnost českých žáků. [online] 2014 [cit 2017-09-24]. Dostupné z: http://www.csicr.cz/Prave-menu/Mezinarodni-setreni/ICILS/Ceska-skolni-inspekce-zverejnuje-vysledky-setreni

BLOCKLY GAMES [online]. Dostupné z: https://blockly-games.appspot.com/

BOBŘÍK INFORMATIKY – Archiv testů. [online]. Dostupné z: https://www.ibobr.cz/test/archiv

BOBŘÍK INFORMATIKY - Informace o soutěži – menu. [online]. Dostupné z: https://www.ibobr.cz/o-soutezi

BROMOVÁ, J. Výuka algoritmizace na ZŠ – současný stav. Diplomová práce. PdF, České Budějovice, 2012.

CODE [online]. Dostupné z: https://studio.code.org/

ČERNÝ, M. Ozobot – malý, ale šikovný. [online] 2014. Dostupné z: http://robodoupe.cz/2014/ozobot-maly-ale-sikovny/

DAUGHERTY, L. Early Education Plays Role in Bridging the 'Digital Divide'. RAND Corporation [online]. 2014. Dostupné z: http://www.rand.org/news/press/2014/03/03.html

DAUGHERTY, L., DOSSANI, R., JOHNSON, E. E., OGUZ, M. Using Early Childhood Education to Bridge the Digital Divide. RAND Corporation [online]. 2014. Dostupné z: http://www.rand.org/pubs/perspectives/PE119.html

EASYSTORE [online]. Dostupné z: https://www.easystore.cz/ozobot-bit-inteligentni-minibot-bily.html#tipy

EURACTIV Děti by se měly učit programovat. Pro jejich zaměstnání to bude nezbytné, tvrdí experti. [online] 15. 10. 2015 [cit. 2017-09-27]. Dostupné z: https://euractiv.cz/section/digitalni-agenda/news/deti-by-se-mely-ucit-programovat-pro-jejich-zamestnani-to-bude-nezbytne-tvrdi-experti-012941/

EURACTIV Programování – Je Česká republika připravená?. [online] 20. 10. 2015 [cit. 2017-09-27]. Dostupné z: http://euractiv.cz/factsheet/obchod-a-export/programovani-je-ceska-republika-pripravena-000136/

FABIAN, D. Podmínky a cykly. [online] 2013. Dostupné z: http://kmlinux.fjfi.cvut.cz/~fabiadav/cecko/poznamky-k-jazyku-c/podminky-a-cykly

FUTSCHEK, G. Algorithmic Thinking: The Key for Understanding Computer Science. In: R.T. Mittermeir (Ed.): ISSEP 2006, LNCS 4226, pp. 159 – 168, 2006. Springer-Verlag Berlin Heidelberg 2006.

FUTSCHEK, G., MOSCHITZ, J. Developing Algorithmic Thinking by Inventing and Playing Algorithms. Constructionism 2010, Paris. [online] 2010. Dostupné z: https://publik.tuwien.ac.at/files/PubDat_187461.pdf

FUTSCHEK, G., MOSCHITZ, J. Learning algorithmic thinking with tangible objects eases transition to computer programming. In International Conference on Informatics in Schools: Situation, Evolution, and Perspectives [online] 2011. Dostupné z: https://publik.tuwien.ac.at/files/PubDat_199953.pdf

GAIO, A. Programming for 3rd graders, Scratch-based or Unplugged? University of Palermo, Department of Mathematics and Computer Science. [online]. Dostupné z: https://keynote.conference-services.net/resources/444/5118/pdf

HLAVENKA, J. Když práce mizí aneb Ničí Internet střední třídu? [online] 7. 6. 2013. Dostupné z: https://www.lupa.cz/clanky/jiri-hlavenka-kdyz-prace-mizi-aneb-nici-internet-stredni-tridu/

HROMKOVIČ, J., KOHN, T., KOMM, D., SERAFINI, G. Examples of Algorithmic Thinking in Programming Education. Olympiads in Informatics, 2016, Vol. 10, 111–124, Vilnius, 2016.

IMAGINE [online] 2005. Dostupné z: http://imagine.input.sk/cz/popis.html

IMAGINE LOGO – Programování pro děti. [online] 2011. Dostupné z: http://www.pf.jcu.cz/imagine/index.php

KALAŠ, I. a kol. Konštrukcionizmus. Od Piageta po školu v digitálnom veku. [online] 2011 [cit. 2016-05-31]. Dostupné z: http://lcd.ent.sk/docs/kalas_a_kol_didinfo2011.pdf

KALAŠ, I., MITTERMEIR, R. Informatics in Schools: Contributing to 21st Century - 5th international conference on Informatics in schools: situation, evolution and perspectives. ISSEP 2011, Bratislava, Slovakia, October 20011, Proceedings

KITCHIN, R. Thinking critically about and researching algorithms. In Information, Communication & Society, 2017, Vol. 20, No. 1, pp. 14–29.

LEGO Mindstorms [online]. [cit. 2017-09-27]. Dostupné z: http://www.lego.com/cs-cz/mindstorms

LESSNER, D. Střípky z konference Didinfo 2015 (2): Výuka informatiky v Polsku. Učíme informatiku [online]. 2015, (1) [cit. 2018-04-30]. Dostupné z: http://ucimeinformatiku.blogspot.cz/2015/06/stripky-z-konference-didinfo-2015-2.html

LITERACY FROM SCRATCH [online]. Dostupné z: http://www.literacyfromscratch.org.uk/

MALÝ, M. Jak naučit děti logickému myšlení a schopnosti řešit problémy? Jde to!. [online] 21. 07. 2017 [cit. 2017-09-03]. Dostupné z: https://www.lupa.cz/clanky/jak-naucit-deti-logickemu-mysleni-a-schopnosti-resit-problemy-jde-to/

MATEMATICKÝ KLOKAN – Informace o soutěži [online]. Dostupné z: http://matematickyklokan.net/index.php/o-soutezi/informace-o-soutezi

McNiff, J. Action research: Principles and Practice. London: Macmillan Education, 1988. Citováno v NEZVALOVÁ, D. Akčním výzkumem k zlepšení kvality školy. e-Pedagogium [online]. 2002, roč. 2, č. 4. [cit. 2017-10-22]. Dostupné z: http://epedagog.upol.cz/eped4.2002/clanek02.htm

MŠMT. Strategie digitálního vzdělávání do roku 2020. [online] 31. 10. 2014. [cit. 2017-07-28]. Dostupné z: http://www.msmt.cz/vzdelavani/skolstvi-v-cr/strategie-digitalniho-vzdelavani-do-roku-2020

MŠ Slovenskej republiky. Vzdelávací program pre 1. stupeň základnej školy pre žiakov so všobecným intelektovým nadaním. ISCED 1 – primarne vzdelavanie. 2009.

NEUMAJER, O. Konference Počítač ve škole 2017: Být digitálně gramotný už neznamená jen ovládat počítač. [online] 27. 4. 2017 [cit. 2017-07-28]. Dostupné z: http://ondrej.neumajer.cz/konference-pocitac-ve-skole-2017-byt-digitalne-gramotny-uz-neznamena-jen-ovladat-pocitac/

NEUMAJER, O. RVP jsou zastaralé. Změnilo se toho tolik, že nemohou odpovídat potřebám digitálních kompetencí. [online] 11. 3. 2017 [cit. 2017-07-28]. Dostupné z: http://www.ceskaskola.cz/2017/03/ondrej-neumajer-rvp-jsou-zastarale.html

OZOBLOCKLY [online] 2017. Dostupné z: https://ozoblockly.com/

OZOBOT - Lesson Library [online]. Dostupné z: https://portal.ozobot.com/lessons

OZOBOT - Basic Training Lesson 1 [online]. Dostupné z: https://portal.ozobot.com/lessons/detail/basic-training-1

OZOBOT Getting Started - Resources and tips to get your classroom ready for Ozobot. [online] 2017. Dostupné z: http://ozobot.com/stem-education/education-getting-started

PAPERT, S. Mindstorm. Children, Computers, and Powerful Ideas. Basic Books. Inc., Publishers : New York, 1980.

PITNER, T. Výuka programování na základní a střední škole. [On-line] [cit. 3.7.2018] Dostupné z: https://www.fi.muni.cz/~tomp/semuc/text_pitner.html

RAMBOUSEK, V. studijní text z předmětu Edukační technologie. 2014 [cit 2017-09-20] Dostupné z: http://moodle.it.pedf.cuni.cz/course/view.php?id=1114

SCRATCH [online]. Dostupné z: https://scratch.mit.edu/

SGP SYSTEM - Výukové programovací nástroje C a C# pro děti, mládež i dospělé. [online] 2016. Dostupné z: http://www.sgpsys.com/cz/

ŠPÚ. Informatika – primárne vzdelávanie.

ŠPÚ. Informatika – nižšie stredné vzdelavanie.

ŠPÚ. Informatika – gymnázium so čtvorročným a patročným vzdělávacím programom.

ŠPÚ. Štátny vzdielavací program. Informatika. Príloha ISCED2. ŠPÚ, 2008.

TIB - Tvorivá Informatika s Baltíkom. [online] 2016. Dostupné z: http://www.tib.sk/

TIŠŇOVSKÝ, P. Scratch: plnohodnotný programovací jazyk nebo jen dětské puzzle? [online] 2011 [cit. 2017-09-27]. Dostupné z: http://www.root.cz/clanky/scratch-plnohodnotny-programovaci-jazyk-nebo-jen-detske-puzzle/

TIŠŇOVSKÝ, P. Seriál Letní škola programovacího jazyka Logo [online] 2007. Dostupné z: http://www.root.cz/serialy/letni-skola-programovaciho-jazyka-logo/#ic=serial-box&icc=title

TOCHÁČEK, D., LAPEŠ, J. Edukační robotika. [online] 2012. Dostupné z: https://kraken.pedf.cuni.cz/~lapej2ap/robo/skripta_edurobo.pdf

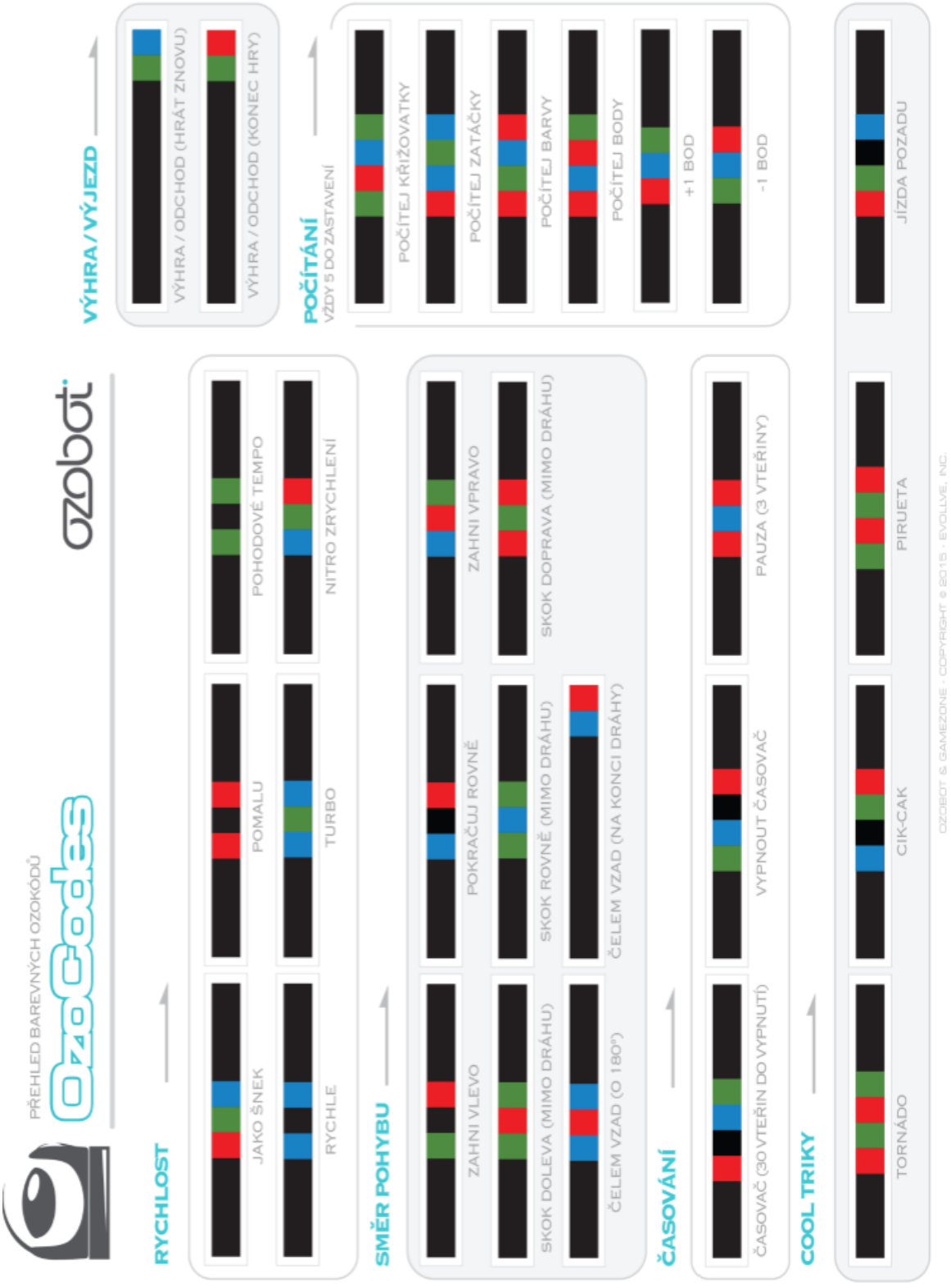
TOMCSÁNYIOVÁ, M. On-line hry pre deti [online]. Dostupné z: http://edi.fmph.uniba.sk/~tomcsanyiova/

WAGNER, J. Bobřík informatiky – výběr úloh z národních kol soutěže 2010 až 2014. Pedagogické.info [online] 14. 3. 2017. Dostupné z: http://www.pedagogicke.info/2017/03/bobrik-informatiky-vyber-uloh-z.html

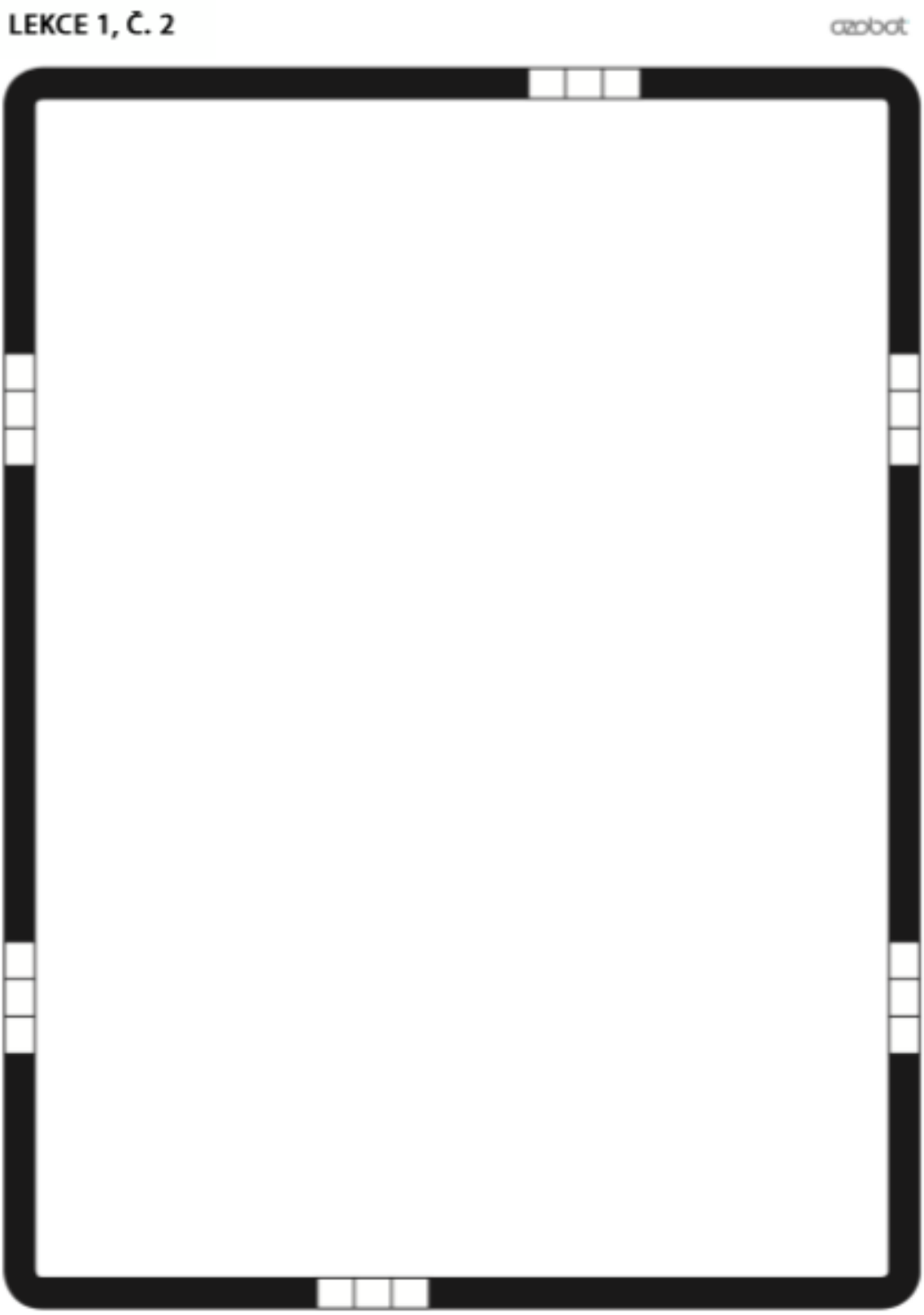
9 Přílohy

Seznam příloh

Příloha 1 – Přehled barevných Ozokódů



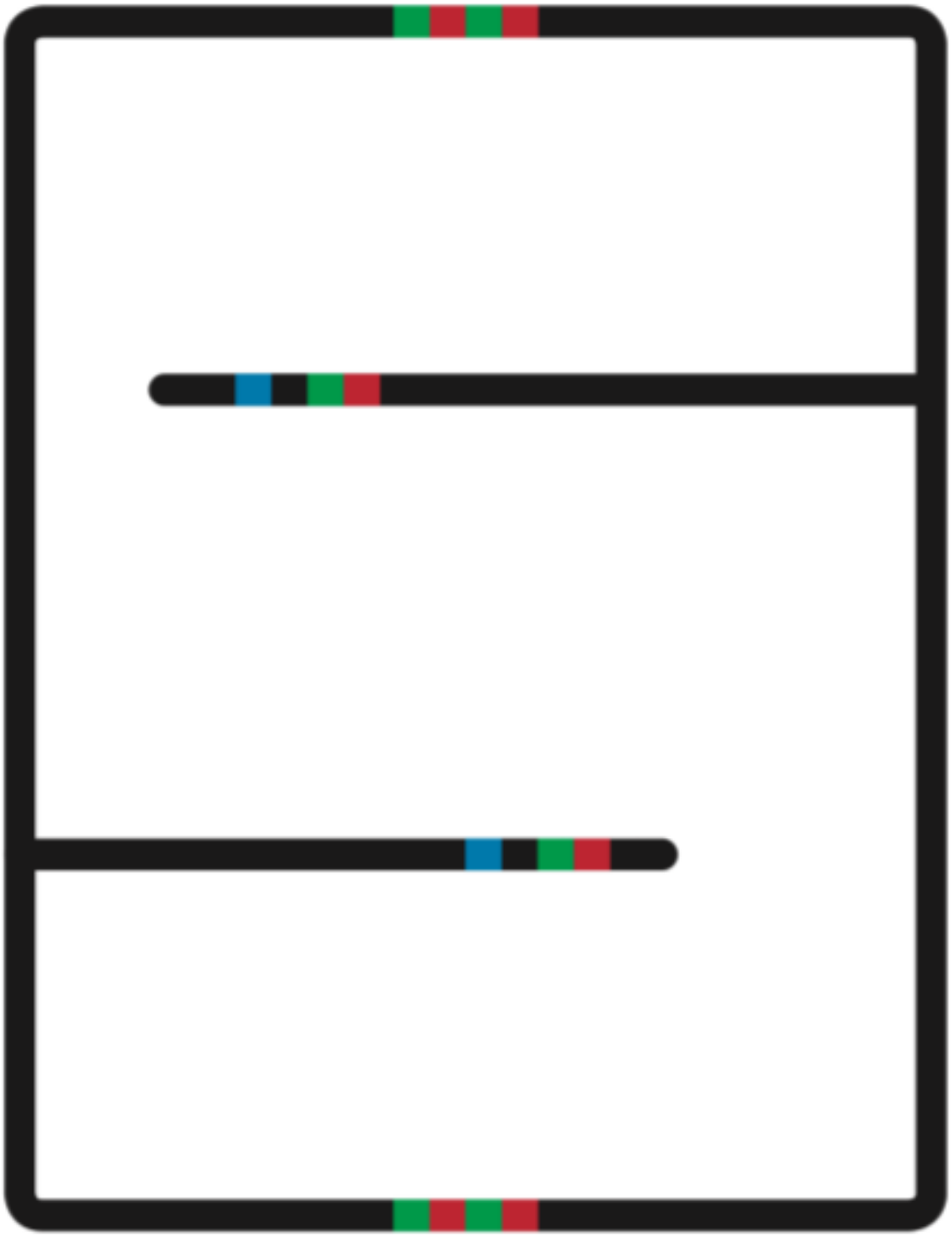
Příloha 2 - Dráha s prázdnými místy pro Ozokódy – Lekce 1, č. 2



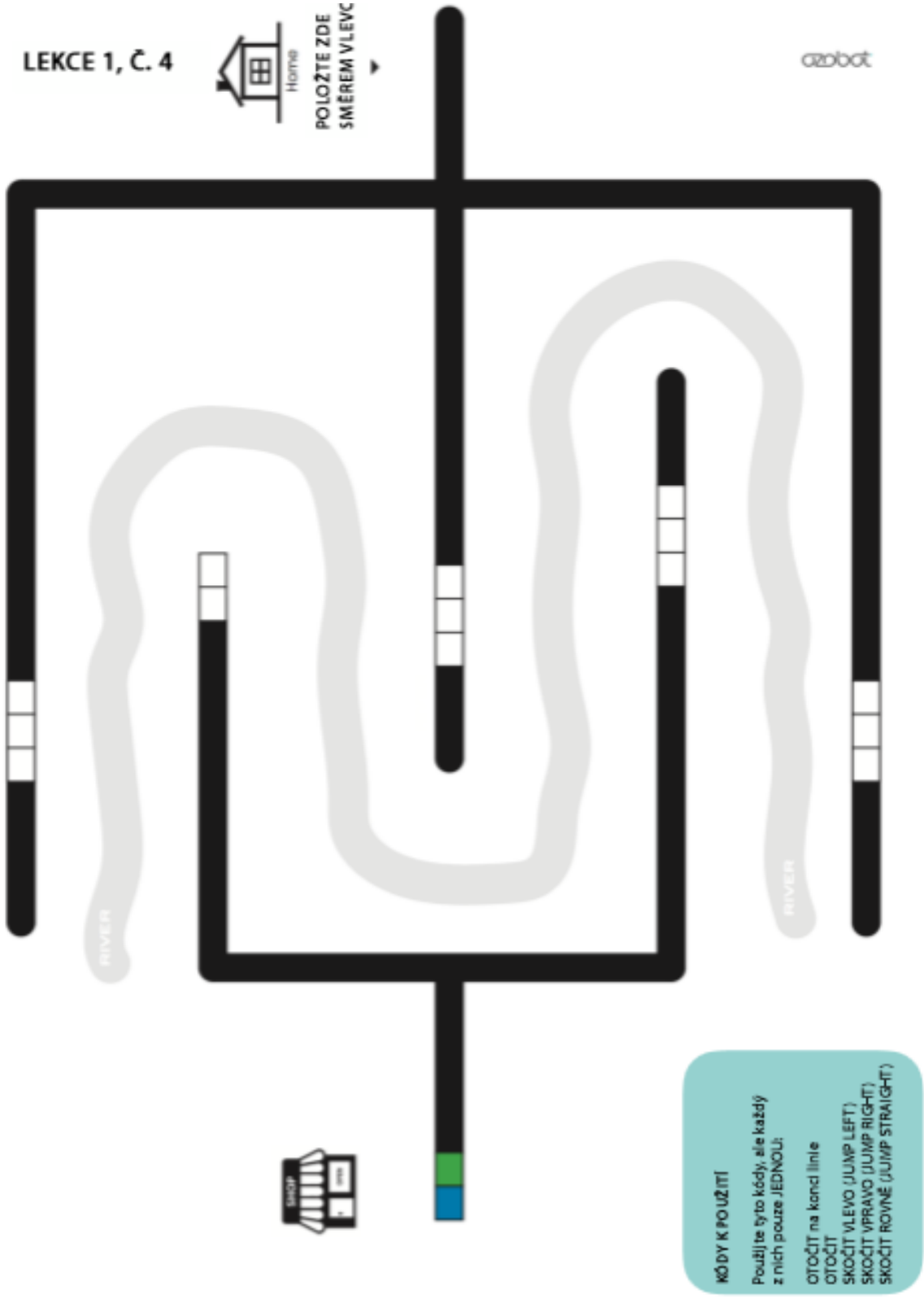
Příloha 3 – Dráha s Ozokódy – Lekce 1, č. 3

LEKCE 1, Č. 3

ozobot



Příloha 4 - Dráha pro úlohou „Cesta k obchodu“ – Lekce 1, č. 4



Příloha 5 – Otázky k navrženým aktivitám s Ozobotem

Přemýšlí robot?

Když se Ozobot pohybuje po červené dráze, tak

Když se Ozobot pohybuje po modré dráze, tak

Když se Ozobot pohybuje po zelené dráze, tak

Jak Ozobot rozpozná barvy?

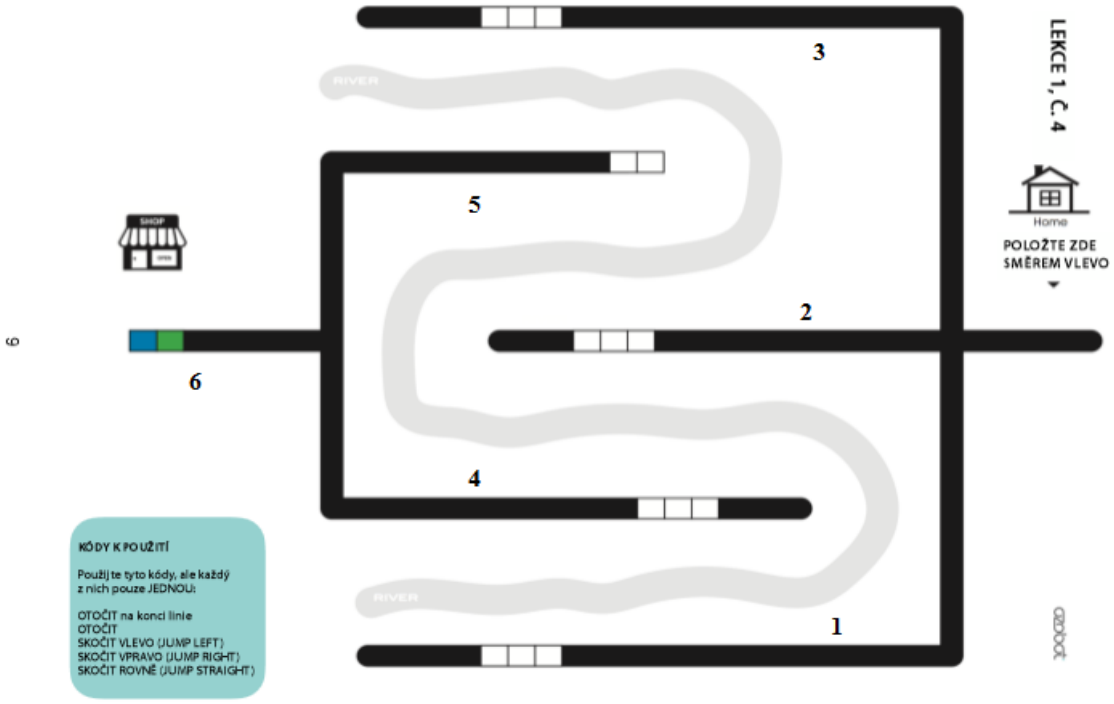
Když Ozobot narazí na pořadí (sekvenci) barev (modrá, **černá**, modrá), tak

Když Ozobot narazí na pořadí (sekvenci) barev (červená, **černá**, červená), tak

Když Ozobot narazí na pořadí (sekvenci) barev (modrá, červená, modrá), tak

Když přijede Ozobot na křižovatku, tak

Když dráha dál nepokračuje (končí), tak Ozobot



Když se na křižovatce vydá touto cestou 1, musí Ozobot

Když se na křižovatce vydá touto cestou 2, musí Ozobot

Když se na křižovatce vydá touto cestou 3, musí Ozobot

Když se na křižovatce vydá touto cestou 4, musí Ozobot

Když se na křižovatce vydá touto cestou 5, musí Ozobot

Příloha 6 – Informovaný souhlas zákonných zástupců žáků

Informovaný souhlas zákonných zástupců žáků o pořízení a využití audio/video záznamu pro potřeby zpracování diplomové práce

Svým podpisem stvrzuji, že souhlasím s účastí svého syna/ své dcery ve výzkumném šetření a s tím, aby učitel Bc. Radek Čuma použil pořízené audio/video záznamy práce mého dítěte pro potřeby zpracování diplomové práce. Šetření se bude zabývat pochopením smyslu a principu použití vybraných algoritmických konceptů při sestavování počítačových programů žáky na 1. stupni základní školy, jelikož výuka programování bude v následujících letech na základních školách povinná. Šetření bude probíhat v rámci výuky předmětu Práce na počítači. Veškerá získaná data budou anonymní, nebudou zveřejněna a budou použita pouze k vyhodnocení výzkumného šetření. Výsledky šetření budou použity k vypracování diplomové práce.

Podpis zákonného zástupce: ……