Univerzita Karlova
Pedagogická fakulta
Katedra informačních technologií a technické výchovy

Diplomová práce

Rozvoj algoritmického myšlení žáků základních škol

Development of algorithmic thinking of primary school pupils

Bc. Jan Vais

Vedoucí práce:	PhDr. Jiří Štípek, Ph.D.
Studijní program:	Učitelství pro střední školy (N7504)
Studijní obor:	Učitelství VVP pro ZŠ a SŠ - informační a komunikační technologie (OKN1IT17)

2021

Prohlašuji, že jsem diplomovou práci na téma Rozvoj algoritmického myšlení žáků základních škol vypracoval pod vedením vedoucího práce samostatně a za použití v práci uvedených pramenů a literatury. Dále prohlašuji, že tato práce nebyla využita k získání jiného nebo stejného titulu.

Praha 19. dubna 2021

. . . . . .

Podpis

Poděkování

S velkým potěšením děkuji vedoucímu diplomové práce PhDr. Jiřímu Štípkovi, Ph.D. a stejně tak i doc. PhDr. Vladimíru Rambouskovi, CSc. za jejich cenné rady, ochotu a trpělivost při vedení této práce. Rád bych také poděkoval vedení vybrané základní školy za možnost uskutečnění akčního výzkumu. V neposlední řadě patří můj dík i celé mé rodině za podporu v závěru studia.

ANOTACE

Předložená diplomová práce zkoumá možnosti rozvoje algoritmického myšlení u žáků základní školy. Algoritmické myšlení představuje nezbytný nástroj k účelné analýze problému a vytvoření postupu k jeho následnému opakovatelnému řešení. Hlavním výzkumným tématem této práce je nalézt efektivní způsoby rozvoje algoritmického myšlení, zejména jak formulovat problém a jak provádět rozvoj algoritmického myšlení s co největším efektem v edukačním procesu.

V práci jsou shrnuty různé způsoby rozvoje algoritmického myšlení a přístupu k jeho výuce. Jsou definovány a analyzovány pojmy informatické myšlení, algoritmické myšlení, algoritmus a algoritmizace. Dále jsou specifikovány soudobé prostředky rozvoje algoritmického myšlení popsané v literatuře. Ty jsou kriticky zhodnoceny a v závěru teoretické části je vybrán nejvhodnější soubor prostředků a navržen způsob jejich aplikace.

Akční výzkum byl realizován v kroužku informatiky na základní škole v Praze. Ověření účinnosti vybraných nástrojů proběhlo v patnácti šedesátiminutových hodinách. Těžiště tohoto výzkumu spočívá ve zvýšení kvality pedagogické praxe vyučujícího, v rozvoji jeho didaktického myšlení i dovedností. Na druhé straně je výsledkem lepší vzdělávání žáků a zdokonalení kvality poskytovaného vzdělávání.

V práci je rozebráno šestnáct lekcí, jsou uvedeny jejich charakteristiky a přínosy v oblasti jednotlivých algoritmických konstruktů. Zkušenosti z jednotlivých úloh jsou shrnuty do metodických poznámek, které usnadňují úlohu pedagoga.

Diplomová práce poskytuje výstupy pro rozvoje algoritmického myšlení. Tyto informace se mohou stát vodítkem a inspirací pro každého učitele usilujícího o rozvoj tohoto myšlení u svých žáků.

KLÍČOVÁ SLOVA

algoritmické myšlení, programovací aplikace, dětské programovací jazyky, algoritmické struktury, aktivity k rozvoji algoritmického myšlení

ANNOTATION

The presented diploma thesis examines the possibilities of developing algorithmic thinking in primary school pupils. Algorithmic thinking is a necessary tool for effective analysis of the problem and the creation of a procedure for its subsequent repeatable solution. The main research topic of this work is effective and efficient ways of developing algorithmic thinking, especially how to formulate a problem and how to develop algorithmic thinking with the greatest effect in the educational process.

The work summarizes various ways of developing algorithmic thinking and the approach to its teaching. The concepts of computer thinking, algorithmic thinking, algorithm and algorithmization are defined and analyzed. Furthermore, contemporary means of developing algorithmic thinking described in the literature are specified. They are then critically evaluated and at the end of the theoretical part the most suitable set of means is selected and the way of their use is proposed.

Action research was carried out in the circle of informatics at the primary school in Prague. The verification of the effectiveness of the selected instruments took place during fifteen sixty-minute lessons. The focus of this research lies in increasing the quality of pedagogical practice of the teacher, in the development of his didactic thinking and skills. On the other hand, it is the result of better education of pupils and improvement of the quality of education provided.

Sixteen lessons are analyzed, their characteristics and benefits in the field of individual algorithmic constructs are presented. Experiences from individual tasks are summarized in methodological notes that enable the role of the teacher.

The diploma thesis provides outputs for the development of algorithmic thinking. This information can become a guide and inspiration for any teacher looking to develop this thinking in their students.

KEYWORDS

algorithmic thinking, programming applications, children's programming languages, algorithmic structures, activities to develop algorithmic thinking

Obsah

1 Úvod

Diplomová práce je zaměřena na zkoumání rozvoje algoritmického myšlení u žáků základní školy. Ústředním tématem zkoumané problematiky je volba a ověření účinných metod pro rozvoj algoritmického myšlení. Práce vychází z předpokladu, že je algoritmické myšlení nepostradatelné pro řešení problémů. Je důležité v rámci výpočetní techniky, ale i v oblasti obecného vzdělání.

V oblasti vzdělávání dostávají stále větší prostor informační a komunikační technologie. Způsob jejich výuky však zůstával v posledních letech stejný. Teprve nyní dochází k revizi ICT kurikula. Stejně jako v ostatních vyspělých státech se snaží Česká republika reagovat na potřeby praxe, která vyžaduje rozvoj informatických kompetencí pro řešení každodenních situací a problémů. Právě rozvoj informatického myšlení by měl být nově implementován do rámcových vzdělávacích programů. Podle připravovaného kurikula by škola neměla v rámci hodin ICT vyučovat používání kancelářských nástrojů (např.: Word, Excel), jak tomu bylo dosud. Nově by žáci prostřednictvím informačních technologií měli rozvíjet schopnost zapojení těchto technologií při řešení nejrůznějších problémů v praxi.[1]

Připravované ICT kurikulum by se mělo obsahově přiblížit úkolům plynoucím ze Strategie digitálního vzdělávání, kterou řeší NÚV (Národní ústav pro vzdělávání) od května 2016.[2] Součástí připravovaných RVP (rámcový vzdělávací program) v oblasti informačních a komunikačních technologií by měly být pojmy jako digitální gramotnost a informatické myšlení. Jednou z hlavních oblastí informatického myšlení je i algoritmické myšlení. Při začlenění do výuky programování na základních školách podporuje rozvoj imaginativního a logického myšlení, které ve výsledku definuje kreativitu člověka.

Algoritmické myšlení je nezbytnou součástí vědeckého pohledu na svět, jeho součástí jsou i obecné myšlenkové schopnosti používané mimo oblast programování například rozdělení problému na dílčí úkoly. Toto myšlení lze chápat jako proces, který vede od

[1] NÚV: *RVP v oblasti Informatiky a ICT* [online]. 2018 [cit. 16. 03. 2019] Dostupné z: `http://www.nuv.cz/t/revize-rvp-ict`.

[2] NÚV, op. cit.

zadání problému až po jeho řešení. V první části je nezbytné porozumění problému a jeho přesné formulace. Dalším krokem je vývoj algoritmu, který umožní problém vyřešit. Nejdůležitější částí je ověření, že vytvořený algoritmus funguje správně, efektivně a pro všechna vstupní data. Teprve poté může být algoritmus implementován a s jeho pomocí získány požadované výsledky.[3]

Na rozdíl od českého vzdělávacího systému se algoritmickým myšlením ve vzdělávání na prvním stupni základních škol zabývají na Slovensku. Žáci tam začínají skládáním hlavolamů či origami podle návodů. Pokračují zápisem vytvořeného postupu. Pomocí počítačů řeší například úlohy na programování robotů. V dětském programovacím prostředí řeší jednoduché algoritmy.[4]

Jak již bylo řečeno, problematika algoritmického myšlení, algoritmizace a programování není součástí učebních osnov (RVP) pro první stupeň základních škol v České republice. Přesto jsou některé školy výjimkou. Vyučují programování například jako volitelný předmět a učí programovací jazyky typu Scratch, Baltika, Imagine Logo atd. Jejich výhodou je, že mohou využívat rodný jazyk žáků a odstraňují tak případnou jazykovou bariéru. Některé základní školy mají programovatelné stavebnice (např. Lego Mindstorms, Merkur, Fischertechnik, Arduino nebo další alternativy).[5]

Výzkumným polem této práce je problematika algoritmického myšlení a možnosti jejího rozvoje na základních školách. Snaží se nabídnout učitelům prostředky k zefektivnění výuky algoritmického myšlení a ukázat jim přínos tohoto myšlení v oblasti klíčových kompetencí žáků pro řešení problémů s využitím výukových softwarů či programovacích jazyků.

Diplomová práce je strukturována do osmi hlavních částí. Úvod práce je zaměřen na vymezení výzkumného pole, cílů a metod práce. V dalších kapitolách je předložen popis

[3] Luay Nakhleh: *What is Algorithmic Thinking?* [online]. 2018 [cit. 09. 08. 2018] Dostupné z: https://www.coursera.org/specializations/computer-fundamentals.

[4] Štátny pedagogicky ústav. *Štátny Vzdelávací Program.* Tech. zpr. 2008. Dostupné z: http://www.statpedu.sk/files/articles/dokumenty/statny-vzdelavaci-program/informaticka_vychova_isced1.pdf.

[5] Hanzalova et al. Algorithm development and programming at elementary education in the Czech Republic. In: *International journal of education and infomation technologies* 9 (2015), s. 175–179. issn: 2074-1316. Dostupné z: http://www.naun.org/main/NAUN/educationinformation/2015/a442008-072.pdf.

zkoumaných termínů a pojmů. V teoretické části jsou popsány a posouzeny jednotlivé způsoby rozvoje algoritmického myšlení. Empirická část práce vychází z poznatků teoretické části a jsou v ní prezentovány průběh a výsledky akčního výzkumu. Poznatky všech částí práce jsou shrnuty v závěrech diplomové práce.

Výsledkem práce je zhodnocení akčního výzkumu a doporučení vhodných metod pro rozvoj algoritmického myšlení na základních školách. Tato práce by měla přispět učitelům k novému pohledu na možnosti rozvoje algoritmického myšlení na základních školách a zároveň napomoci ke zvýšení jeho efektivity.

2 Vymezení výzkumného pole

2.1 Výzkumné problémy

Hlavním výzkumným problémem diplomové práce je zkoumání způsobů rozvoje algoritmického myšlení u žáků základní školy. Lze jej formulovat do otázky, jak provádět rozvoj algoritmického myšlení s co největším přínosem v edukačním procesu.

Z hlediska definovaného problému lze vyčlenit dílčí výzkumné problémy formulované do následujících otázek:

P₁ Jaký význam má rozvoj algoritmického myšlení pro žáky základní školy?

P₂ Jaké prostředky, nástroje či způsoby lze využít pro rozvoj algoritmického myšlení?

P₃ Které z těchto prostředků jsou vhodné pro rozvoj algoritmického myšlení u žáků základní školy?

P₄ Jak tyto prostředky umožňují rozvíjet a prohlubovat algoritmické myšlení?

2.2 Cíle a úkoly práce

Hlavním cílem diplomové práce je identifikovat a deskribovat efektivní způsoby rozvoje algoritmického myšlení. Tento hlavní cíl je dále členěn cíli dílčími.

C₁ Definovat a analyzovat pojmy: informatické myšlení, algoritmické myšlení, algoritmus a algoritmizace. Následně specifikovat jejich význam pro žáka.

C₂ Analyzovat soudobé prostředky rozvoje algoritmického myšlení.

C₃ Zhodnotit jednotlivé prostředky rozvoje algoritmického myšlení a stanovit nejvhodnější.

C_4 Prozkoumat vliv aplikování vybraných prostředků rozvoje algoritmického myšlení na žáka a navrhnout nejlepší způsoby jejich využití.

2.3 Výzkumné metody

Z hlediska metodologie budou pro dosažení stanovených cílů použity teoretické a empirické metody. Z teoretických metod bude uplatněna metoda terminologické a obsahové analýzy pojmů: algoritmus, algoritmizace, algoritmického a informatického myšlení ve směru naplnění cíle $\mathbf{C_1}$. Pro splnění cíle $\mathbf{C_2}$ bude aplikována analýza informačních zdrojů, která bude založena na studiu primárních a sekundárních pramenů a jejich interpretaci. Aplikována bude také komparativní analýza způsobů rozvoje algoritmického myšlení ve směru naplnění cíle $\mathbf{C_3}$. Dále proběhne posouzení vhodnosti využití jednotlivých prostředků.

Teoretická východiska plynoucí z cíle $\mathbf{C_3}$ umožní ověřit empirický výzkumný projekt založený na základě kvalitativních metod pedagogického výzkumu. Projekt je jako akční výzkum[1] zaměřený na zkoumání zařazení jednotlivých způsobů rozvoje algoritmického myšlení do výuky na základní škole. Úkolem výzkumu je ukázat možnosti přímého nasazení vybraných metod. Důvodem pro volbu akčního výzkumu je rovnocenné postavení výzkumníka - učitele i zkoumaných žáků. Všichni mají stejný podíl na vyhodnocení a zkoumání výsledků, což by mohlo usnadnit nasazení jednotlivých metod rozvoje algoritmického myšlení do současné praxe. Akční výzkum bude využit k naplnění cíle $\mathbf{C_4}$.

[1] Jan Hendl. *Přehled statistických metod.* 4., rozš. Praha: Portál, 2012. ISBN: 978-80-262-0200-4.

Část I

Teoretická východiska práce

3 Základní pojmy

Kapitola vymezuje základní pojmy v oblasti informatického resp. algoritmického myšlení tak, jak budou chápány pro účely této práce.

Jedním z hlavních úkolů informatického vzdělávání je spojovat algoritmické myšlení s dalšími druhy myšlení jako je logické, vědecké, inovativní a efektivní. Všechny zmíněné druhy myšlení lze shrnout pod pojmem informatické myšlení.

3.1 Informatické myšlení

Informatické myšlení představuje Lessner[1] jako schopnost myslet při řešení problému jako informatik. Podle Wingové[2] patří informatické myšlení mezi základní dovednosti stejně jako čtení, psaní a počítání. Z obou názorů však vyplývá, že výuka informatiky a případně i výuka programování neslouží pouze ke schopnosti tvořit programy. Cílem je zařadit dovednost informatického myšlení mezi ostatní dovednosti důležité pro praktický život.

O informatickém myšlení hovořil již Papert[3] od začátku devadesátých let. I ve svých dalších publikacích popisuje přínosy využití počítačů při výuce matematiky tak, aby bylo rozvíjeno informatické myšlení.[4]

V literatuře můžeme najít několik definic informatického myšlení. Například u autorů Barr a Stephenson[5] je možné najít základní definici informatického myšlení přehledně

[1] Daniel Lessner. Analysis of Term Meaning Computational Thinking. In: *Journal of Technology and Information* 6.1 (2015), s. 71–88. ISSN: 1803537X. DOI: 10.5507/jtie.2014.006, s. 72.

[2] Jeannette M Wing. Computational Thinking. In: *Commun. ACM* 49.3 (2006), s. 33–35. ISSN: 0001-0782. DOI: 10.1145/1118178.1118215. Dostupné z: http://doi.acm.org/10.1145/1118178.1118215, s. 33.

[3] Seymour Papert et al. Situating constructionism. In: *Constructionism*. 1991, s. 15.

[4] Seymour Papert. An exploration in the space of mathematics educations. In: *International Journal of Computers for Mathematical Learning* (1996). ISSN: 13823892. DOI: 10.1007/BF00191473. Dostupné z: http://www.papert.org/articles/AnExplorationintheSpaceofMathematicsEducations.html.

[5] By Valerie Barr et al. Bringing Computational Thinking To K12. In: 2.1 (2011), s. 48–54. ISSN: 21532184. DOI: 10.1145/1929887.1929905, s. 52.

v tabulce pro různé obory vzdělávání.

Furber[6] v kapitole 3 zprávy Královské společnosti píše definici informatického myšlení jako proces rozpoznávání jednotlivých informatických aspektů světa kolem nás a aplikování počítačových nástrojů a technik k porozumění přirozeným i umělým systémům a procesům.

Podle Kempa[7] je důležité, že informatické myšlení je základem pro studium ICT. Dále zdůrazňuje přesah mimo samotnou práci s počítačovými technologiemi. Umožňuje problémy řešit, vyřazovat ty neřešitelné a vymýšlet algoritmy pro jejich řešení. Souhrnně lze říci, že informatické myšlení zahrnuje: rozpoznávání a rozklad problému, abstraktní myšlení, zobecňování problémů a návrh algoritmu.

Rozklad je rozdělení složitého problému na menší části, které jsou snáze srozumitelné. Tyto části pak mohou být jednodušeji řešeny. Po rozkladu jsou často nalezeny podobnosti mezi jednotlivými dílčími rozloženými problémy, které nám pomohou vyřešit složitější problémy efektivněji. Abstraktní myšlení pomáhá použít pouze detaily, které jsou nezbytné pro řešení problému. Zobecnění řešení problému dává šanci tato řešení opakovaně použít. Výsledkem informatického myšlení by měl být algoritmus jako postupně seřazené instrukce nezbytné pro řešení problému.

Ze všech přístupů vyplývá, že informatické myšlení je využitelné téměř v jakémkoliv vzdělávacím předmětu a v praktickém životě.

3.2 Algoritmické myšlení a jeho význam

Algoritmické myšlení je soubor schopností, které souvisí s konstrukcí a porozumění algoritmům. Zahrnuje schopnost analyzovat problém a přesně ho specifikovat, schopnost nalézt základní kroky a pomocí nich vytvořit správný algoritmus, který bude schopen problém vyřešit. Součástí těchto dovedností je i schopnost přemýšlet o všech aspektech

[6] Simon Furber. Shut down or restart? The way forward for computing in UK schools. In: *Technology* January (2012), s. 1–122, s. 29.

[7] Peter Kemp. *Computing in the national curriculum A guide for secondary teachers.* Tech. zpr. 2014, s. 34, s. 5.

algoritmů včetně zlepšování jeho efektivity. Toto myšlení má velký kreativní aspekt, který je využit při konstrukci nových algoritmů.[8]

Na základních školách je nejvýhodnější začít řešením problémů, které nejsou závislé na konkrétním programovacím jazyce. Právě začátečníci by mohli mít problémy s pochopením programovacího jazyka. Bylo by pro ně bylo těžké, soustředit se na návrh algoritmu. Pro žáky, kteří zatím nemají žádné zkušenosti, je možné použít například pseudokód. Jedná se o neformální přepis z programovacího jazyka do přirozené řeči. Problémy, které by měli žáci řešit, by měly být snadno srozumitelné, ale ne příliš jednoduché. Složitější problémy, které vyžadují například neobvyklá řešení, poskytují více prostoru pro zapojení kreativity.[9]

3.3 Algoritmus

Pod pojmem *algoritmus* můžeme chápat množinu jasně definovaných instrukcí určujících pořadí provedení konečného počtu základních kroků, které zajišťují vyřešení všech úloh daného typu v konečném čase.[10]

[8] Gerald Futschek. Algorithmic Thinking: The Key for Understanding Computer Science. In: *Lecture Notes in Computer Science 4226* (2006), s. 159–168. Dostupné z: `https://publik.tuwien.ac.at/files/PubDat_140308.pdf`, s. 160.

[9] Ibid.

[10] Milan Hejný Jozef Hvorecký Beloslav Riečan Znám Štefan, Bukovský Lev. *Pohľad do dějín matematiky.* Bratislava: Alfa, 1986, s. 181–182.

3.3.1 Vlastnosti algoritmů

Algoritmus je definován základními vlastnostmi:[11]

Elementárnost Skládá se z konečného počtu základních kroků. Tyto kroky lze snadno realizovat.

Determinovanost Každý krok algoritmu je jasně určen a definován. Po provedení jednoho kroku lze určit jaký další krok následuje, případně zda má algoritmus skončit.

Konečnost Algoritmus musí skončit. Jinak řečeno, má konečný počet elementárních kroků, které vedou k výstupu. Tento počet kroků může být libovolně velký.

Rezultativnost Algoritmus vede vždy alespoň k jednomu výsledku.

Hromadnost Pomocí daného algoritmu je řešena obecná skupina problémů, nikoliv jeden konkrétní problém.

Efektivnost Někteří autoři, např. Skalka a kol.[12] uvádějí ještě tuto šestou vlastnost. Algoritmus proběhne v co nejkratším čase a s využitím co nejmenšího množství prostředků.

Splnění těchto vlastností je velmi důležité proto, že v informatice vykonávají algoritmy především „nemyslící“ počítače. Stroj si nedokáže uvědomit, že výpočet trvá podezřele dlouho, nedokáže experimentovat, nemá žádné zkušenosti a nedokáže se poučit z vlastních chyb. Proto je důležité, aby algoritmus tyto základní vlastnosti splňoval.

Elementárnost

Každý postup může být zapsaný více způsoby. Při jeho navrhování je třeba dbát na to, aby všechny instrukce byly srozumitelné, jednoduché a jednoznačné. Pokud je vyu-

[11] Miroslav Virius. *Základy algoritmizace*. Dot. 1. vy. Praha: České vysoké učení technické, 1997. ISBN: 80-01-01346-4. Dostupné z: `http://www.rudisweb.wz.cz/dokumenty/algoritmizace.pdf`, s. 9.

[12] Ján Skalka et al. *Algoritmizácia a úvod do programovania*. Nitra: UNIVERZITA KONŠTANTÍNA FILOZOFA V NITRE, 2007, s. 158. ISBN: 9788080942175, s. 11.

čována algoritmizace, pak je třeba zvažovat, které příkazy jsou žákům srozumitelné.[13] Například při pohybu herní postavy je vhodnější používat otáčení pomocí příkazů *otoč se vpravo* a *otoč se vlevo* než *otoč se o 90stupňů* nebo *otoč se o -90stupňů*.

Pokud je algoritmus určený pro žáky, pak je to výhoda. Člověk se dokáže učit na základě předchozích zkušeností snadněji než počítač. Přesto je třeba si při formulaci instrukcí dávat pozor na jednoznačnost.

Determinovanost

Klade na algoritmus požadavek, aby bylo v každém kroku jasné, jakým směrem má řešení problému pokračovat. Člověk by možná dokázal pochopit, jak pokračovat díky vlastním zkušenostem, ale pro počítač musí být posloupnost kroků jednoznačná. Logickým důsledkem této vlastnosti je, že při stejných vstupních podmínkách bude výsledek pokaždé stejný.[14]

Konečnost

Splnění této vlastnosti má zajistit, aby počet kroků byl konečný. Pokud s problémem pracuje člověk, zpravidla dokáže určit, zda neskončil v nekonečné smyčce. Počítač se bez zkušeností na stejné úrovni rozhodnout nedokáže. Kromě nekonečných problémů se můžeme setkat i s řešeními, která jsou sice konečná, ale trvají velmi dlouho. Typickým příkladem jsou šifrovací algoritmy, kde lze teoreticky rozšifrovat každou zprávu, ale doba realizace je tak dlouhá, že smysl zprávy po rozšifrování ztrácí smysl.[15]

Rezultativnost

Tato vlastnost od algoritmu vyžaduje, aby jeho realizace prokazatelně vedla po konečném počtu kroků minimálně k jednomu výsledku. Při použití nesprávného algoritmu

[13] Ibid.
[14] Skalka et al., op. cit., s. 12.
[15] Skalka et al., op. cit., s. 13.

obvykle získáme také výsledek, který může být pro daný algoritmus správný. Problém je ale v tom, že tento algoritmus neřeší danou úlohu.[16]

Hromadnost

Pokud je od algoritmu vyžadováno, aby byl hromadný, je potřeba, aby bylo možné měnit vstupní parametry. Ne každý algoritmus ale umí být hromadný. Některé algoritmy řeší konkrétní problém a není u nich možné vstupní hodnoty měnit. Proto je tato vlastnost považována spíše za užitečnou než nezbytnou.[17]

Efektivnost

Vytvořit efektivní algoritmus znamená, navrhnout takový postup, který vyřeší problém s použitím minimálního množství prostředků a v co nejkratším čase. I neefektivní algoritmus je algoritmem, ale z pohledu zadavatele je efektivnost velmi důležitá zvláště při zpracování velkého množství údajů. Při složitých problémech je obvykle prvotním cílem vytvoření algoritmu, ale po jeho otestování může být požadována jeho optimalizace.[18]

Pro účely této práce je pravděpodobně dostačující základní definice algoritmu jako postupu řešení problémů v několika na sebe navazujících krocích. Tento popis algoritmu lze jednoduše aplikovat na každodenní činnost člověka. Velmi běžným příkladem je recept na výrobu palačinek.

3.4 Algoritmizace

Prvotním důvodem vytváření algoritmů je existence problému, který je potřeba vyřešit. Zjednodušeně je možné říci, že celý život je plný problémů, které je třeba řešit. Přestože je pro proces algoritmizace důležité znát několik definic, je asi tím nejdůležitějším odrazovým můstkem schopnost analyzovat a myslet. Tyto dovednosti vedou k tomu,

[16] Skalka et al., op. cit., s. 12.
[17] Skalka et al., op. cit., s. 14.
[18] Ibid.

aby byla složitá úloha rozdělena na jednodušší. Umožňují lepší pochopení podstaty problému.[19]

Ve škole se často učí převážně teoretickým poznatkům. Žáci znají pravidla, věty, axiomy, ale často nevědí, co s nimi dělat v průběhu řešení problému. Neznalost postupů či metod usuzování jim brání dojít k hledanému výsledku. Pokud je vyučující schopen algoritmizovat výuku, tedy vytvořit pro žáky takový algoritmus, který je dovede k cíli, pak má jeho výuka mnohem lepší výsledky.[20]

Pokud chce vyučující algoritmizovat učební látku, musí nejdříve definovat cíl vyučování. To znamená čemu chce žáka naučit, jaké vědomosti, dovednosti a návyky u něho chce vytvořit. Dalším krokem je nalezení způsobu zjišťování osvojení probírané látky. Častou chybou je předpoklad, že žák správně zvládl látku, odříká-li definici. To ale neznamená, že pojem ovládá. Je potřeba ověřit, že žák umí pojem používat. V další etapě je třeba definovat požadavek na výsledný stav a zároveň popis vstupního stavu. Pokud se má například žák naučit anglické číslovky od jedné do deseti, musíme definovat v kolika procentech případů dokáže odpovědět nejenom správným vyslovením číslovky, ale i v předem definovaném reakčním čase. Zcela analogicky můžeme popsat stav těchto hodnot na začátku vyučování. Pokud o anglických číslovkách nic neví, pak jsou hodnoty správnost i čas rovny nule. Úkolem učitele je převést žáka od vstupního stavu do hodnot stavu výstupního. Teprve dosažením výstupních hodnot můžeme danou úlohu pokládat za vyřešenou.[21]

Abychom mohli zkonstruovat algoritmus výuky, je třeba rozčlenit vyučování na řadu jednotlivých kroků, tedy řadu dílčích algoritmů. Tyto algoritmy je pak třeba poskládat do vhodné posloupnosti a definovat ty druhy úloh, které zajišťují úspěšný přechod z jedné potřebné operace do druhé. Je nutno dát žákovi úkol a instrukci, jak ho má plnit. Podstata algoritmizace vyučovacího procesu spočívá v tom, že žák dosáhne výstupního stavu, ať už bude reagovat nejrůznějšími možnými způsoby. Vyučující musí brát předem v potaz všechny možné reakce, vědět, jak na různé kroky žáků reagovat tak, aby byli

[19] Skalka et al., op. cit., s. 9.
[20] L N Landa. *Algoritmy a učení*. Vyd. 1. Praha: Státní pedagogické nakladatelství, 1973, s. 94.
[21] Landa, op. cit., s. 196–205.

dovedeni k požadovanému stavu.[22]

3.5 Algoritmické struktury

Pro dorozumění se s počítačem je třeba použít jednoznačných příkazů. Standardně představuje příkaz jednu elementární činnost, kterou je schopen vykonavatel algoritmu realizovat. Vykonávání příkazů ve stejném pořadí v jakém jsou zapsány, se označuje *sekvence*. Pokud vytváříme komplexní algoritmy, je třeba uvést jejich vstupní a výstupní podmínky a zároveň označit začátek a konec algoritmu.[23]

V některých případech je třeba zajistit vykonání příkazu jenom při splnění definovaných podmínek. Možnost rozhodnout se a vykonat příkaz na základě pravdivosti zkoumaných podmínek se označuje jako *větvení*. Skládá se z podmínky a příkazů, které se vykonají v případě kladného nebo záporného výsledku. Pokud je podmínka splněna, pokračuje se vykonáváním příkazů v kladné větvi. V opačném případě se zpracovávají příkazy ve větvi záporné.[24]

Velmi často je potřeba část algoritmu zopakovat. Zápis umožňující opakování se jmenuje cyklus. Při každém opakování je důležité, co se má opakovat (tělo cyklu) a do kdy se má opakovat (podmínka cyklu). V závislosti na vztahu mezi tělem a podmínkou cyklu se dělí na: *cyklus s předem známým počtem opakování, cyklus s podmínkou na začátku a cyklus s podmínkou na konci.*[25]

Sekvence, větvení a *cyklus* se nazývají algoritmické konstrukce a platí, že každý algoritmus je zapsán jejich vhodnou kombinací. Nejvhodnější pro začátečníky je skládat algoritmy prostřednictvím grafických bloků.

[22] Landa, op. cit., s. 206–220.
[23] Skalka et al., *Algoritmizácia a úvod do programovania*, s. 18.
[24] Skalka et al., op. cit., s. 23.
[25] Skalka et al., op. cit., s. 31.

4 Způsoby rozvoje algoritmického myšlení

Z počátku se může žákům zdát pojem algoritmus jako příliš abstraktní. Učitel jim může pomoci různými metodami k tomu, aby práci s algoritmy lépe pochopili. Jedním ze způsobů je představení pojmu formou hry. Úkolem učitele je poskytnout správné informace o problému a položit návodné otázky tak, aby žáci přemýšleli o vytvoření algoritmu, který tyto problémy vyřeší. Zároveň by měl učitel žáky motivovat k tomu, aby jejich řešení bylo co nejefektivnější. Právě vytváření algoritmů je vhodnou příležitostí k pohybovému zapojení žáků, kteří si mohou při hraní algoritmy přímo zažít.[1]

4.1 Přístupy k výuce algoritmického myšlení na základní škole

Přestože existuje všeobecný názor o tom, že algoritmy jsou používány pouze při řešení matematických a informatických problémů, je důležité představit žákům algoritmy nejprve pomocí příkladů z jejich denního života. Pokud však výuka probíhá na prvním stupni ZŠ je otázkou, zda vůbec slovo algoritmus před žáky zmiňovat.[2]

Klasický způsob výuky, kdy se žáci učí například definice, je zde pravděpodobně zbytečný. Role učitele zde dostává novou podobu průvodce, který žákovi pomáhá a zasahuje pouze v případě, kdy žák neví, jak dále pokračovat.[3]

Algoritmické myšlení lze rozvíjet na základní škole pravděpodobně prostřednictvím všech školních předmětů.

[1] Gerald Futschek et al. Developing Algorithmic Thinking by Inventing and Playing Algorithms. In: *Constructionism* (2010), s. 1–10, s. 1.

[2] J. Mezak et al. Learning scenarios and encouraging algorithmic thinking. In: *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2018 - Proceedings* (2018), s. 760–765. doi: 10.23919/MIPRO.2018.8400141, s. 4.

[3] Ibid.

V České republice se teprve chystá změna kurikula resp. RVP. Předpokládá se, že po jejich změně bude výuka algoritmického myšlení promítnuta do výuky informačních technologií prostřednictvím ŠVP.[4]

Na Slovensku je v kurikulárních dokumentech jediným cílem předmětu informatika uvažování v algoritmech, nalézání algoritmických řešení problémů, vytváření návodů a programů podle daných pravidel.[5]

Podle tohoto dokumentu žák na konci 4. ročníku ZŠ umí v oblasti algoritmického řešení problémů:[6]

- řešit problémy přímým řízením vykonavatele např. robota;
- aplikovat elementární příkazy daného jazyka (ze slovníku příkazů) na řízení vykonavatele;
- realizovat návod, postup, algoritmus řešení úlohy - interpretovat ho, navrhnout řešení krok po kroku a simulovat činnost vykonavatele.

Další schopností slovenského čtvrťáka by mělo být vyhledávání chyb ve výsledných algoritmech. Tuto chybu opravit v zápisu řešení a o těchto řešeních diskutovat.[7]

Pojetí informatiky v Polsku je pravděpodobně také na vyšší úrovni, než v současné době u nás. Programování ve výuce je zde použito jako nástroj ke splnění dalších cílů, například rozvoje algoritmického myšlení. Cílem výuky není probírat programovací jazyky, ale porozumět informatice.[8]

Cílem výuky předmětu Computing v Anglii je podle národního kurikula získání kom-

[4] NÚV, *RVP v oblasti Informatiky a ICT*.

[5] Štátny pedagogicky ústav. *Informatika - primárne vzdelávanie*. Tech. zpr. Štátny pedagogický ústav, 2014, s. 1–10. Dostupné z: http://www.statpedu.sk/sk/svp/inovovany-statny-vzdelavaci-program/inovovany-svp-1.stupen-zs/matematika-praca-informaciami/, s. 2.

[6] Štátny pedagogicky ústav, op. cit., s. 7.

[7] Štátny pedagogicky ústav, op. cit., s. 7–8.

[8] Daniel Lessner: *Střípky z konference Didinfo 2015: Výuka informatiky v Polsku* [online]. 2015 [cit. 23. 03. 2019] Dostupné z: http://ucime-informatiku.blogspot.com/2015/06/stripky-z-konference-didinfo-2015-2.html.

petencí problémy analyzovat a zapisovat řešení v podobě počítačového programu.[9]

Ve věkové kategorii 7-11 let by žáci měli umět v programech použít *posloupnost*, *výběr* a *opakování*, pracovat s různými vstupy a výstupy. Také by měli být schopni na základě logického uvažování vysvětlit, jak pracují jednoduché algoritmy. Měli by umět odhalit a opravit chyby v programech a algoritmech.[10]

4.2 Prostředky a nástroje využívané pro rozvoj algoritmického myšlení

Algoritmizaci je možné vyučovat v různých prostředích. Tato prostředí lze rozdělit do 5 základních skupin, které jsou rozvedeny v následujících kapitolách. Jednotlivé skupiny můžeme posuzovat z několika hledisek. Tato práce je hodnotí podle typů aktivit, které je možné využívat. Druhý pohled se zaměřuje na algoritmické struktury a jejich využití.

4.2.1 Unplugged metody

Unplugged metody jsou založené na konstruktivistickém přístupu. Studenti dostávají úkoly založené na jednoduchých pravidlech a při řešení těchto problémů se sami přirozeně propracovávají problematikou. Při tomto způsobu výuky si žáci uvědomují, že dobré nápady jsou na dosah. Jakmile zjistí, jak tento způsob výuky funguje, začnou vidět další možnosti řešení.[11]

Tyto metody neučí žáky programovací jazyky, ale rozvíjí a procvičují algoritmické myšlení, které je pro programování nezbytné. Žáci mohu v klidu bez využití výpočetní tech-

[9] Department for Education. The national curriculum in England: Framework document. In: *Department for Education* December (2014), s. 264. ISSN: 17532167. Dostupné z: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/381344/Master_final_national_curriculum_28_Nov.pdf, s. 230–233.

[10] Miles Berry. *Computing in the national curriculum: A guide for primary teachers*. 2013. ISBN: 9781783391431. Dostupné z: http://www.computingatschool.org.uk/data/uploads/CASPrimaryComputing.pdf, s. 6.

[11] CS Unplugged: *CS Unplugged* [online]. 2020 [cit. 05. 11. 2020] Dostupné z: https://csunplugged.org/en/how-do-i-teach-cs-unplugged/.

niky prozkoumávat své vlastní myšlenkové postupy díky spolupráci a zapojení vlastního těla.[12] Žáci se stávají součástí problému a musí sami přispět k jeho vyřešení.[13]

Unplugged metody nám zajišťují, že se žáci učí hrou, zvykají si na algoritmické myšlení, které si mohou ověřovat metodou pokus-omyl. Většiny těchto metod se účastní skupiny žáků, protože kooperují a přispívají tím i k rozvoji spolupráce a komunikace. Velkou výhodou je malá náročnost na vybavení, nezávislost na prostředí a výpočetní technice.[14]

Typické pro unplugged metody je vedení žáků k tomu, aby prostřednictvím her a soutěží sami objevovali řešení. Pokud jsou během aktivit použity drobné hmotné předměty pak může vznikat ještě zábavnější atmosféra, která žáky více motivuje. Velmi často se pro pobavení žáků používá příběh, který obsahuje záhadné či tajemné prvky.[15]

Unplugged metody se zaměřují na výuku základních algoritmických principů.

4.2.2 Programovatelní roboti

Další oblastí, která podporuje výuku algoritmického myšlení, jsou programovatelní roboti. Žáci zde mají možnost programovat a hlavně ověřovat funkčnost zvoleného řešení. Velmi dobrým spojením konstrucionistických forem vzdělávání s technologiemi jsou programovatelní roboti. Způsoby ovládání robota jsou různé, od nejjednodušších, kde se tlačítka nacházejí přímo na robotovi, přes složitější, když robot sleduje čáry a barevné kódy, až po nejsložitější programování v programovacím jazyce.

Robota mohou žáci prostřednictvím příslušného softwaru řídit a ovládat. Program zpravidla nabízí základní grafické prostředí pro chování robota.

[12] Thomas J. Cortina. Broadening participation: Reaching a broader population of students through "Unplugged" activities. In: *Communications of the ACM* 58.3 (2015), s. 25–27. ISSN: 15577317. DOI: 10.1145/2723671.

[13] Aubrey Lawson. Pedagogy of CS Unplugged : Lessons from Outreach and Education Activities in Computer Science. In: ACM (2017). Dostupné z: https://pdfs.semanticscholar.org/1b18/2ed67a35bc413b34db77df1876c6a367ccb5.pdf.

[14] Tomohiro Nishida et al. New methodology of information education with computer science unplugged. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 5090 LNCS (2008), s. 241–252. ISSN: 03029743. DOI: 10.1007/978-3-540-69924-8_22, s. 28.

[15] Nishida et al., op. cit., s. 29–30.

Ve vzdělávací sféře se nachází několik druhů robotických systémů využitelných pro podporu výuky.

Bee-Bot

Pro nejmenší žáky z 1. stupně základních škol je ideální použití robota ve vzhledu včely. Bee-Bot má na vrchní části směrová tlačítka, která mu naprogramují pohyb vpřed, vzad, vlevo a vpravo. Po stisknutí zeleného tlačítka mohou žáci sledovat průběh jimi zadaného programu. Pomocí světel a zvuků poznají dokončení každého kroku. Maximální počet příkazů, které lze uložit do paměti je 40.[16]

Ozobot

Ozobot je malý robot ve tvaru koule o průměru cca 3 centimetry. Na povrchu má pouze jediné tlačítko, ale programovat ho je možné několika způsoby. První způsob programování je pomocí barevné sekvence. Základní barvou pro Ozobota je černá, červená, modrá, zelená a bílá. Barvy Ozobot sleduje při svém pohybu pomocí optického senzoru a představují příkazy, který ozobot po načtení vykoná. Druhý způsob programování je v prostředí OzoBlockly. Tento způsob programování má několik úrovní obtížnosti. Je možné ho využít na základní i střední škole. K programování s OzoBlockly potřebujeme počítač, tablet nebo smartphone. Třetím způsobem ovládání Ozobota jsou speciální aplikace pro tablety dostupné v Google Play nebo App Store.[17]

Na trhu se nachází starší verze Ozobot Bit a novější Ozobot Evo. U modelu Evo dochází k maximálnímu využití potenciálu Ozobota až prostřednictvím aplikací.

Pro rozvoj algoritmického myšlení je výhodnější využít Ozo kódy. Žáci si s nimi zkouší základní principy programování a neruší je problémy spojené s načítáním nedokonale zakreslených kódů.

[16] TTS Group Ltd. *Bee-Bot Rechargeable, chield friendly, programmable fllor robot.* Nottinghamshire: TTS Group Ltd., 2015, s. 8. Dostupné z: `https://resources.terrapinlogo.com/robots/bee-bot/user_guide_and_technical_specs.pdf`.

[17] Inc. Ozobot and Evollve: *Ozobot* [online]. 2020 [cit. 14. 03. 2021] Dostupné z: `https://ozobot.com`.

Nevýhodou programování pomocí ručně kreslených barev může být nedůslednost žáků při ručním kreslení.

Qobo

Robot Qobo je ideální pro předškoláky. Představuje ho motorizovaný šnek, který se pohybuje po barevných kartičkách. Každá z nich představuje jednu instrukci a jimi se robot řídí. Najdeme zde směr jízdy, svícení či troubení. Výhodou je, že v základním balení je také dobíjecí USB kabel a 30 základních karet. Pokud dítě zvládne základní sadu, je možné buď karty dokupovat nebo použít virtuální v programu ScratchJr a využít USB kabel.[18]

Matatalab

Jinou variantou robota je Matatalab. Ten se pohybuje podle kostiček seřazených na speciální desce. Tyto pokyny přečte speciální pozorovací věž. Děti poskládají kostičky představující různé operace, věž je přečte a vyšle pokyny robotovi. I zde je možné dokupovat rozšiřující komponenty.[19]

Boffin

Pro starší děti je určena stavebnice Boffin a její novější verze Boffin Magnetic. Na elektromagnetické stavebnici je možné postavit jednoduchý obvod i s displejem. Součástky, které děti využijí jsou rezistor, kondenzátor, tranzistor apod. Učí je základům elektroniky.[20]

[18] Robobloq Co. Ltd.: *Qobo* [online]. 2020 [cit. 14. 03. 2021] Dostupné z: `https://www.robobloq.com/product/Qobo`.

[19] Matatalab: *matatalab* [online]. 2020 [cit. 14. 03. 2021] Dostupné z: `https://matatalab.com/en`.

[20] BOFFIN: *Boffin magnetic* [online]. 2020 [cit. 14. 03. 2021] Dostupné z: `https://boffinmagnetic.com/cs`.

Q-Dino

Dalším programovatelným robotem je Q-Dino. Dinosaurus se nápadně podobá stavebnici Merkur a je prostřednictvím Bluetooth a mobilní aplikace Robobloq programovatelný. Fyzické kartičky jsou zde nahrazeny softwarovými bloky. Děti nejsou omezeny základní sadou a případným dokupováním dalších karet. Robot je programovatelný v dětském programovacím jazyku Scratch. Poté co děti přestane bavit programování ve Scratchi, mohou začít používat Python nebo dokonce C++.[21]

Micro:bit

Existují i náročnější roboti. Jedním z nich je například britský Micro:bit. Jedná se o skutečný mikropočítač se vstupy, výstupy, malou základní deskou s USB, která obsahuje navíc Bluetooth a sadu LED tlačítek. V nejnovější verzi najdeme i mikrofon a reproduktor. Základní desku lze rozšířit o další moduly. Robot je opět programovatelný v jazyku Scratch, pro pokročilejší i v jazyce Python.[22]

4.2.3 Robotické stavebnice

Tyto stavebnice se osvědčují při výuce tím, že hravou formou rozvíjí představivost, tvořivost, technické myšlení a v návaznosti na programování robotů i algoritmické myšlení. Potenciál v dětské robotice zasahuje do mnoha odvětví. Hodně se zde projevuje teorie konstrukcionismu Seymoura Paperta, kdy žák je maximálně zapojen do všech činností. Na rozdíl například od dětských programovacích jazyků si musí dítě samo nejdříve stavebnici zkonstruovat, tedy vyřešit první problém. Pak následují další činnosti. Po celou dobu tam mají žáci možnost objevovat, tvořit, programovat a ověřovat výsledky svého konání. Výhodou robotických stavebnic je spojení stavebnice se softwarovou podporou.

[21] Robobloq Co. Ltd.: *Q-Dino* [online]. 2020 [cit. 14. 03. 2021] Dostupné z: `https://www.robobloq.com/product/Q-dino`.

[22] Educational Foundation micro:bit: *Micro:bit* [online]. [cit. 14. 03. 2021] Dostupné z: `https://microbit.org/`.

Mindstorms

K nejvýznamnějším zástupcům programovatelných stavebnic patří sady ze série LEGO Mindstorms. Základem jsou standardní LEGO Technic součástky. Navíc jsou dodávány další funkční prvky jako jsou senzory či programovatelné řídící kostky. Dále je možné implementovat další speciální produkty externích výrobců. Žáci si mohou sestavovat modely podle vlastní fantazie a využívat různé míry složitosti týkající se vzhledu modelu jeho činnosti, řízení či ovládání. Pro potřeby žáků je možné zakoupit jednotlivé sady již obsahující jednotlivé díly potřebné pro sestavení plně funkčního robota. Tyto sady jsou prodávány ve speciálním režimu pro školy a vzdělávací instituce v plastových boxech. Nejnovější prodávaná sada této série je LEGO Mindstorsm EV3.[23]

We-Do

Jedná se o jednodušší verzi Lego Mindstorms. Hlavním rozdílem jsou menší možnosti při sestavování robota. Jeho řídící jednotka obsahuje pouze dva vstupy a nabízí pouze dva senzory, jeden pohybový a senzor náklonu. Tato stavebnice je primárně určena pro předškolní děti. Jejím hlavním cílem je seznámit je se světem robotiky a základy programování. I přes všechna omezení lze ze stavebnice postavit mnoho různých robotů.

I zde lze využít programovací jazyk Scratch. Propojení robota a řídící jednotky je zajištěno pomocí Bluetooth technologie.[24]

Stavebnice VEX IQ

Stavebnice se skládá z plastových destiček propojených kolíčky nebo osičkami. Pro pohyb jsou využívány motory a ozubená kolečka či gumové pásy. Stavebnice je řízena prostřednictvím programovatelné kostky. Jedná se o velkou centrální jednotku s baterií, displejem a několika porty. Její baterie napájí všechny součásti, které potřebují energii

[23] LEGO® Education: *LEGO MINDSTORMS* [online]. 2020 [cit. 14. 03. 2021] Dostupné z: https://www.lego.com/cs-cz/themes/mindstorms/about.

[24] LEGO® Education: *WeDo 2.0* [online]. 2020 [cit. 14. 03. 2021] Dostupné z: https://education.lego.com/en-us/products/lego-education-wedo-2-0-core-set/45300#confidence.

a umožňuje přístup uživateli a spuštění připravených programů. Jednotlivé součásti se připojují přes síťové kabely. Jediné co je možné použít bezdrátově, je ovladač. Po krátkém propojení s kostkou je možné ho nadále používat bezdrátově.[25]

Stavebnice Merkur

I s touto stavebnicí se děti mohou stát velkými staviteli a zároveň programátory. Prostřednictvím systému Merkur Open Source lze všechny stavebnice rozšiřovat o další a další prvky. Řídící deska obsahuje zdrojový kód, kterým je jízda po černé čáře. Stavebnice je v základu dodávána včetně motoru a řídící jednotky s procesorem. V základu je i modul infra, který je možné vhodnou změnou zdrojového kódu libovolně naprogramovat. Vše je kompatibilní s PC a manuál i programy jsou dodávány na CD. Přestože výrobce nabízí dokoupení dalších dílů, je třeba počítat s tím, že na robotech není moc místa k připojení dalších komponent.[26]

4.2.4 Aplikace pro rozvoj algoritmického myšlení

Existuje několik her, které rozvíjejí algoritmické myšlení u žáků a zároveň je učí základům programování. Jedná se zpravidla o hry, ve kterých žáci ovládají postavu pomocí příkazů. To nejjednodušší je ovládání pomocí šipek. O něco složitější jsou hry, kde žáci využívají místo šipek příkazy na obrázku. Příkazy se vykonávají buď hned nebo až po zmáčknutí speciálního tlačítka.

GalaxyCodr

Galaxycodr je interaktivní vzdělávací hra, díky které se žáci zábavným způsobem učí algoritmickému myšlení a základním principům programování. Prostředí hry představuje galaxie, která se skládá z několika planet. Na každé planetě se žák může naučit

[25] Inc. VEX Robotics. *VEX IQ - Uživatelská příručka řídicího systému.* Dostupné z: `https://www.veskole.cz/downloads/VEX/VEX_IQ_Uivatelska_piruka_idiciho_sytemu.pdf`.

[26] Merkur Toys s.r.o.: *Robotika |Merkur* [online]. [cit. 07. 03. 2021] Dostupné z: `https://eshop.merkurtoys.cz/robotika-c10/`.

něco nového.[27] Hrdina musí díky šikovnosti žáka překonávat různé nástrahy, které na něj čekají na cestě za záchranou svého nejlepšího kamaráda. Hra je založena na logickém myšlení žáků, kteří musí poskládat jednotlivé bloky s příkazy tak, aby hlavní hrdina přešel určenou trasu. Ve hře žáci sbírají za správně splněnou úlohu body. Při určitém součtu nasbíraných bodů jsou odměněni kartičkou s vyobrazením nového obyvatele jedné z planet.[28]

BotLogic

Hra BotLogic je zasazená do dvourozměrného labyrintu. Pohybuje se v něm robot a cílem je dostat robota do domečku pomocí příkazů nahoru/dolu, vpravo/vlevo. V dalších úrovních se přidávají další příkazy, například zmáčkni. Ve hře je omezený počet příkazů, ale ve vyšších úrovních si je schopen robot přidávat příkazy sbíráním baterek.[29]

Zvláštností je, že se v horní části obrazovky objevuje kromě posloupnosti jednotlivých kroků i textová podoba příkazů. Pokud jsou hráči zdatnější, mohou tam přímo dopisovat příkazy místo obrázkových tlačítek. Syntaxe příkazů je poměrně jednoduchá, takže ji mohou zkušenější hráči využívat. Může pro ně být dobrým začátkem pro využití textové formy příkazů. Hra rozvíjí algoritmické myšlení a navíc je každá úroveň sledována časomírou, která působí motivačně.

Lightbot

Lightbot je plnohodnotná programovací aplikace. Hráč zde ovládá robota, který rozsvěcuje žárovky. Pohybuje se po čtvercové ploše ve všech čtyřech směrech. Využívá příkazy dopředu, otoč se vpravo/vlevo, vyskoč a rozsviť. Všechny příkazy jsou v obrázkové podobě a hráč je skládá na předem připravená místa. Teprve po zmáčknutí tlačítka start,

[27] Galaxycodr.com: *Galaxycodr.com* [online]. 2017 [cit. 20. 08. 2019] Dostupné z: `https://www.facebook.com/pg/galaxycodr/about/`.

[28] Portál Skolske.sk: *Princípy programovania sa môžu učiť už druháci a tretiaci na ZŠ* [online]. 2017 [cit. 20. 05. 2020] Dostupné z: `https://www.skolske.sk/clanok/32395/principy-programovania-deti`.

[28] https://www.botlogic.us

[29] BotLogic.us Team: *Botlogic.us* [online]. 2019 [cit. 03. 03. 2021] Dostupné z: `https://botlogic.us/about`.

robot vykoná to, co bylo naprogramováno.[30]

Ve hře jsou tři úrovně: základy, procedury a cykly. Pro každou úroveň je k dispozici jen omezený počet příkazů, žáci se tak učí i efektivitě. Ve vyšších úrovních přicházejí náročnější úkoly a možnost použít dvě pomocné procedury. Hráč si musí dopředu promyslet strategii tak, aby mu vystačily přidělené příkazy. Hra je přiměřeně náročná pro první stupeň základních škol.[31]

Blockly games

Je název pro grafické programovací prostředí vyvinuté společností Google. Jedná se o vizuální jazyk, který pomáhá rozvíjet algoritmické myšlení. Nepotřebuje tedy velké vysvětlování či speciální pokyny. Žáci se zde seznamují s instrukcí, cykly, procedurami, funkcemi a proměnnými. Tento dětský programovací jazyk přibližuje žáky k jazyku JavaScript. Ten se objevuje jako výsledek po úspěšně splněné úloze.[32]

Run Marco

Je aplikace kompletně přeložená do několika jazyků včetně češtiny. Úkolem Marca nebo Sophie je dostat se stanovenou trasou do cíle. Postupně se zde objevují překážky.[33]

Robomise

Jedná se o simulátor, ve kterém se žáci učí programovat prostřednictvím blokového programování. Cílem je dostat raketu do modrého pásma. Výhodou této aplikace je, že nabízí stále těžší a těžší úlohy. Celkem má 9 úrovní, každá se skládá ze tří různě náročných fází.[34]

[30] LightBot Inc: *LightBot* [online]. 2017 [cit. 11. 10. 2020] Dostupné z: `https://lightbot.com`.

[31] LightBot Inc, op. cit.

[32] Google Inc.: *Blockly-games* [online]. 2020 [cit. 11. 07. 2020] Dostupné z: `https://blockly.games`.

[33] Run Marco: *Run Marco* [online]. [cit. 20. 06. 2019] Dostupné z: `runmarco.allcancode.com`.

[34] MUNI FI: *RoboMise* [online]. [cit. 01. 07. 2019] Dostupné z: `https://robomise.cz`.

CodeCombat

Je výborně propracovaná herní aplikace z fantasy prostředí. Je o něco složitější z hlediska grafiky a pravidel. Funkce postavy, kterou si hráč zvolí, jsou prostřednictvím splněných úkolů rozšiřovány. Grafika je poněkud složitá, ale žákům pomáhají nápovědy a instrukce. Ty se objeví právě v okamžiku, kdy si hráč neví rady nebo udělá chybu. Veškeré akce postavy jsou zapsány kódem. Pro psaní kódu je možné si zvolit jazyk Python, JavaScript, CoffeeScript a Lua.[35]

Made with code

Opět se jedná o programování pomocí bloků. Aplikace se zaměřuje více na děvčata. Tato aplikace vznikla poté, co vlastní výzkum společnosti Google odhalil, že je třeba podpory mladým ženám, aby se stejně jako chlapci i ony věnovaly počítačové vědě.[36]

4.2.5 Dětské programovací jazyky

Na podporu rozvoje algoritmického myšlení na prvním stupni základní školy se používají dětské programovací jazyky. Podle Ľ. Salanciho[37] jsou to většinou vizuální jazyky přizpůsobené této věkové kategorii. Jsou jednoduché, srozumitelné a zároveň pro žáky dostatečně atraktivní. Jsou tak odlišné od programovacích prostředí pro programátory.

Výhodou jejich použití k výuce základů programování a rozvoje algoritmického myšlení je jejich jednoduchá syntaxe. Žáci používají pouze příkazy.

Některé z těchto jazyků mají navíc ještě zjednodušenou formu. Například Logo má Easy Logo a Scratch má ScratchJr. To ulehčuje těm nejmladším se také zapojovat a na základě zkušeností pak přejít na vyšší verzi programovacího jazyka. Tyto jazyky se vy-

[35] CodeCombat Inc.: *Code Combat* [online]. [cit. 25. 06. 2019] Dostupné z: `https://codecombat.com`.

[36] Google Inc.: *Code with google* [online]. [cit. 01. 03. 2020] Dostupné z: `https://edu.google.com/code-with-google/`.

[37] Ľubomír Salanci. *Didaktika programovania*. České Buďějovice: Jihočeská univerzita v Českých Buďějovicích, Pedagogická fakulta, 2018, s. 32. Dostupné z: `https://imysleni.cz/images/vyukove_materialy/JU_Didaktika_PRG.pdf`, s. 22.

značují tím, že nabízejí mnoho smysluplných a atraktivních úloh s rostoucí náročností. To posiluje motivační faktor učení. Tím nejvyšším stupněm by měl být volný mód, při kterém již žáci vytváří vlastní programy bez předpřipraveného zadání.

Mezi nejznámější dětské programovací jazyky patří Baltík, Logo / EasyLogo a Scratch / ScratchJr.

Baltík

Baltík je dětský programovací jazyk, který zahrnuje několik režimů. Nejdříve se žáci seznámí s režimem „skládat scénu“. Tam se seznamují s prostředím a pomocí myši skládají scénu do čtvercové mřížky. Tento režim je ten nejjednodušší a zatím se nejedná o programování.[38]

Druhý režim „vyčaruj scénu“ už je přípravou na programování. Žák v něm ovládá čaroděje Baltíka, který se může pohybovat dopředu/vpravo/vlevo a umí vykouzlit předměty.

Třetí režim je již čistě programátorský a má dvě části pro začátečníky a pokročilé. Zde už žáci mohou tvořit program pomocí kartiček. Tady už je jich daleko více, například mohou kroky opakovat či spojovat příkazy do bloků. Program se spustí tlačítkem spustit a celý vytvořený program se animuje. Kromě celého programu se dá spustit jen označený blok příkazů.

Poslední režim pro pokročilé programování již nabízí možnosti běžné pro vyšší programovací jazyky jako jsou proměnné, náhodná čísla či podmínky. Tato část je vhodná pro starší a zdatnější žáky.

EasyLogo

EasyLogo obsahuje dva módy, jeden s předpřipravenými aktivitami a druhý kreativní. V prvním módu je 25 předpřipravených aktivit. Žáci se nejdříve seznamují s prostředím a ovládáním postavičky pomocí třech příkazů - dopředu, otoč se vpravo/vlevo. Zde se

[38]SGP: *Multimediální tvůrčí systém SGP Baltík 3* [online]. [cit. 04. 03. 2021] Dostupné z: `https://www.sgpsys.com/cz/DescriptionB3.asp`.

příkazy vykonávají hned po zmáčknutí příslušné ikonky a žák vidí ihned výsledek svého programu.[39]

Další části jsou již věnovány programování. Příkazy se posouvají do samostatného sloupce a u každého se nechají měnit parametry, nastavit počet kroků dopředu nebo úhel otočení postavy. Příkazy se vykonávají ihned po jejich přidání do programu, dají se vymazat, upravovat nebo měnit jejich pořadí.[40]

Některé aktivity učí žáky používat cykly, jiné například obsahují předpřipravený kód, ve kterém je chyba. Žáci mají program upravit tak, aby byl výstup správný. V poslední části žáci tvoří procedury, tedy úlohy, které na sebe navazují. V kreativním módu už mohou programovat volně. EasyLogo je vhodné pro žáky, kteří budou potom navazovat programováním ve vyšší verzi Logo.

Logo

Je poměrně starý projekt, přesto se udržel po mnoha vylepšeních až do dnešních dní. Součástí robota je i pero, které může zaznamenávat jeho trasu. První Logo se objevuje již v 60. letech 20. století. Bylo spojeno s robotem představujícím želvu. Ta se pohybuje pomocí tří atributů: umístění, směr a pero. Programovací jazyk Logo je také důvodem pro vznik názvu „želví grafika“.[41]

Dnes je dětský programovací jazyk Logo i po téměř čtyřiceti letech stále využíván pro výuku algoritmizace. Není už jen svázaný s původním tvarem želvy, ale malování zůstalo. Na obrázku, který žák prostřednictvím Loga namaloval, je na první pohled vidět, zda je program dobře napsaný. K atraktivnosti jazyka přispívá to, že s elementární znalostí programování a geometrie, dokáže vytvářet složité a zajímavé vektorové obrázky. Dnes je již želva převážně virtuální a pohybuje se i ve více rovinách.

[39] Ľubomír Salanci. *EasyLogo – discovering basic programming concepts in a constructive manner*. 2010, s. 10. Dostupné z: http://www.salanci.sk/EasyLogo/Paper.pdf, s. 3.

[40] Ibid.

[41] Seymour Papert. The Children's Machine: Bringing the Computer Revolution to Our Schools, Seymour Papert. In: *Bulletin of Science, Technology and Society* 14.4 (1994), s. 227–227. ISSN: 0270-4676. DOI: 10.1177/027046769401400426. Dostupné z: http://journals.sagepub.com/doi/10.1177/027046769401400426.

ScratchJr

Je zjednodušenou formou vyšší verze Scratch. Opět je to verze věnovaná těm nejmenším. Jsou dostupné bezplatné aplikace na tablety (Android i iOS). Žáci mají k dispozici několik pozadí, která se dají upravovat a několik postav a objektů, které mohou programovat pomocí kartiček. Celkem mají k dispozici 6 skupin příkazů: podmínky, příkazy související s pohybem postavy, s postavou (mluví, zvětší se/zmenší se, je ne/viditelná), zvukové možnosti, nastavení rychlosti a opakování či ukončení příkazů.[42]

Pro každou postavičku lze poskládat několik typů událostí. Žáci mohou vytvářet několik typů událostí ve spodní části obrazovky, po kliknutí na zelenou vlajku se program spustí na celou obrazovku.

Jedná se o prostředí vhodné pro školy, které mají k dispozici tablety pro žáky na nižším prvním stupni a očekává se, že budou dále programovat ve Scratchi.

Scratch

Je jednoduchý programovací jazyk, který mohou využívat i žáci prvního stupně základních škol. Posloupnost kódu se vytváří sestavováním připravených bloků za sebou. Výhodou je, že je možné si stále z příkazů vybírat a soustředit se na problém samotný. Nenutí žáky umět kódovat v textovém jazyce. Scratch je volně dostupný v online verzi.[43]

Jedná se o ideální edukační nástroj vhodný pro rozvoj kompetencí k řešení problémů rozvoji algoritmického myšlení. Programovací koncepty korespondují s běžně využívanými programovacími koncepty: sekvence, cyklus, podmínky, proměnná, vlákna, události, náhodná čísla atd.[44]

[42] MIT Media Lab et al.: *About ScratchJr* [online]. 2017 [cit. 03. 03. 2021] Dostupné z: `http://www.scratchjr.org/about/info`.
[43] MIT: *Scratch* [online]. 2019 [cit. 12. 10. 2019] Dostupné z: `https://scratch.mit.edu`.
[44] MIT, op. cit.

KODU Game Lab

Je programovací jazyk vyvinutý společností Microsoft. Celý systém je postaven na výběru příkazu z nabídky nástrojů na kruhových výsečích. Každá nabídka v sobě obsahuje několik konkrétních nástrojů. Jednotlivé řádky programového kódu se již podobají klasickému programování, jsou například uvozeny podmínkovým příkazem. Žáci se zde setkávají s řízeným průběhem programu.[45]

4.3 Zvolené prostředky rozvoje algoritmického myšlení

Cílem výukových lekcí je zařadit takové aktivity, které žákům rozvíjí algoritmické myšlení. Podle revidované Bloomovy taxonomie jsou ideální pro rozvoj algoritmického myšlení ta nejvyšší poznání a zároveň s nimi dojít až do dimenze tvorby.[46] V ideálním případě by žáci po absolvování kroužku měli být schopni uplatnit získané strategie i v dalším řešení problémů. Z tohoto pohledu by bylo nejlepší použít takové aktivity, ve kterých by pak sami mohli pokračovat bez návodných zadání. Aby to bylo možné, musí tomu předcházet fáze, ve které žáci získávají tyto strategie. Je ideální použít takové nástroje, které žáky provedou od nejjednoduššího až po složitější s tím, že na konci pololetí budou schopni aplikovat získané strategie i v reálném životě.

[45] Microsoft Research: *Kodu Game Lab* [online]. 2020 [cit. 04. 03. 2020] Dostupné z: `http://www.kodugamelab.com`.

[46] Vladimír Rambousek. *Vybrané kapitoly z didaktiky a psychodidaktiky.* Praha: Univerzita Karlova v Praze, Pedagogická fakulta, 2014, s. 73. ISBN: 978-80-7290-671-0. Dostupné z: `https://uprps.pedf.cuni.cz/UPRPS-476-version1-30_rambousek.pdf`, s. 15–21.

Dílčími cíli je vyhodnotit použití jednotlivých aplikací pro rozvoj algoritmického myšlení. Zjistit jaké aplikace jsou nejvhodnější. Zda jsou dostatečně motivující, zda žáky základní školy baví, zda mají snahu dané problematice porozumět. Všechny cíle stanovené pro výzkumné šetření navazují na cíle $\mathbf{C_3}$ a $\mathbf{C_4}$. Výzkumné otázky pro realizovaný akční výzkum:

- Které z vhodných nástrojů mají největší motivační úspěch u žáků programovacího kroužku?
- Jakými způsoby podporují jednotlivé nástroje rozvoj algoritmického myšlení?
- Mají některé metody omezení ve vztahu ke zkoumanému vzorku žáků?
- Je možné, aby žáci po absolvování kurzu dále rozvíjeli algoritmické myšlení pomocí vlastních zadání?
- Který z nástrojů lze doporučit jako nejvýhodnější podle obou předchozích hledisek?

4.3.1 Výběr aktivit k rozvoji algoritmického myšlení

Celkem bylo v teoretické části zmíněno 5 kategorií rozvoje algoritmického myšlení. Všechny metody by se za jedno pololetí v programovacím kroužku nedalo použít. Je třeba udělat výběr a vybrat 1 - 2 metody, které by pro 4. až 5. ročník základní školy byly co nejvhodnější.

Jako první byly uvedeny unplugged metody. Tyto metody jistě mají své místo při rozvoji algoritmického myšlení. Naše zkušební skupina však bude věkově neheterogenní tým. Za těchto podmínek se může stát, že věkově starší a zkušenější členové týmu budou mít velký zájem na hledání a nalezení řešení, a proto se může stát, že se mladší členové nezapojí a do herních principů proniknou jen málo nebo vůbec. Je možné, že by tyto aktivity podporovaly rozvoj algoritmického myšlení spíše u aktivnějších členů týmu. Jako nevýhoda použití unplugged metod se také jevila realizace akčního výzkumu v rámci kroužku programování, protože žáci očekávali maximum práce s počítačem.

Přestože jsou unplugged aktivity velkým pomocníkem v rozvoji algoritmického myšlení, do tohoto výzkumu nebudou zařazeny.

Roboti a robotické stavebnice jsou rovněž určeny pro týmovou spolupráci, platí pro ně stejná nevýhoda pro věkově nestejnorodé skupiny. Navíc zde objevuje další nevýhoda a to je vysoká cena. Jak roboti, tak i robotické stavebnice jsou cenově velmi náročné. Proto ani tyto aktivity nebudou do akčního výzkumu zařazeny.

4.3.2 Výběr z aplikací a dětských programovacích jazyků

Pro výběr těch nejvýhodnějších aktivit byla zvolena komparativní analýza. Cíle byly stanoveny, ale aby bylo možné jednotlivé metody porovnat, je třeba stanovit kritéria:

K1 vhodnost pro věkovou kategorii (obtížnost) 10 - 11 let - váha 6;

K2 motivační a atraktivní charakter pro danou věkovou skupinu - váha 3;

K3 rozvoj algoritmického myšlení - váha 6;

K4 technická (jazyk, technické možnosti - počítače v dané škole) a finanční dosažitelnost - váha 9.

Jednotlivé aplikace a dětské programovací jazyky jsou hodnoceny v tabulkách 4.1 a 4.2. Každá kategorie je hodnocena známkou 1 - 5 přičemž 1 znamená nejvhodnější a 5 nejméně vhodné. Hodnocení každé aplikace je posuzováno individuálně podle zkušeností a znalostí autora. Pro každou aplikaci a programovací jazyk je spočítán vážený průměr v závislosti na důležitosti každého kritéria.

Vzhledem k tomu, že základem pro použití jednotlivých aplikací je nezbytná technická dosažitelnost v dané škole, byla tomuto kritériu **K4** přidělena nejvyšší váha 9. V tomto kritériu je zároveň obsažena i finanční dosažitelnost.

V dalších dvou kritériích se objevuje vhodnost pro danou věkovou kategorii (**K1**) a možnosti rozvoje algoritmického myšlení v kritériu **K3**. Pro tyto dvě kritéria byla stanovena váha 6. Jedná se o třetinu nižší váhu, než předchozí kritérium **K4**.

Váha 3 byla přidělena kritériu **K2**, které řeší, jak velký motivační charakter má aplikace pro danou věkovou kategorii. Jedná se o nejnižší váhu. Všechna kritéria a jejich váhy byly stanoveny dle vlastních zkušeností autora práce.

Tabulka 4.1: Porovnání vhodnosti jednotlivých aplikací

Váhy kritérií / Aplikace	K1 6	K2 3	K3 6	K4 9	Vážený průměr
Blockly games	2	3	1	1	1,50
Botlogic	3	2	2	2	2,25
CodeCombat	4	1	2	1	2,00
GalaxyCodr	1	1	2	1	1,25
Lightbot	2	2	1	1	1,38
Made with code	1	4	3	1	1,88
Robomise	3	2	1	1	2,63
Run Marco	2	2	2	4	2,75

Tabulka 4.2: Porovnání vhodnosti jednotlivých dětských programovacích jazyků

Váhy kritérií / Aplikace	K1 6	K2 3	K3 6	K4 9	Vážený průměr
Baltík	4	4	3	3	3,38
EasyLogo	3	3	2	2	2,38
Logo	3	3	1	2	2,13
KODU Game Lab	3	1	2	5	3,25
ScratchJr.	4	2	2	4	3,25
Scratch	1	1	1	1	1,00

Z tabulky porovnání 4.1 jednotlivých aplikací vyplývá, že:

- Největší motivační úspěch u žáků programovacího kroužku se předpokládá u aplikací CodeCombat a Galaxy Coder.
- Největší potenciál pro rozvoj algoritmického myšlení mají hry Blockly Games, Ligtbot a Robomise.
- Věkové omezení najdeme u aplikace CodeCombat a technické omezení (aplikace je pouze pro tablety) u hry Run Marco.
- Po srovnání všech hledisek a započtení jejich váhy bude nejvýhodnější použití aplikací BlocklyGames, GalaxyCoder a Lightbot.

V každém případě je třeba začít s jednoduššími programovacími konstrukty a pokračovat směrem ke složitějším.

Z tabulky 4.2 vyplývá, že:

- U programovacího jazyka KODU Game a Scratch se předpokládá největší motivace žáků.
- Programovací jazyky Logo a Scratch mají zajímavý potenciál pro zvyšování algoritmického myšlení u žáků dané věkové kategorie.
- Kritérium pro technickou a finanční dosažitelnost nejlépe splňuje programovací jazyk Scratch.
- Základní předpoklad věkové hranice 10 – 11 let splňuje pouze dětský programovací jazyk Scratch.
- Po využití váženého průměru se jeví jako nejlepší použití dětského programovacího jazyka Scratch.

5 Návrh uceleného souboru aktivit orientovaný na rozvoj algoritmického myšlení u žáků 4. a 5. tříd

Z tabulky 4.1 vyplývá, že nejvhodnějšími nástroji rozvoje algoritmického myšlení pro danou věkovou kategorii žáků základní školy by mohly být aplikace GalaxyCoder, Lightbot a Blockly Games. Z kategorie dětských programovacích jazyků podle tabulky 4.2 se jeví jako nejvhodnější jazyk Scratch. V následujících kapitolách budou aktivity s jednotlivými aplikacemi popsány podrobně.

5.1 Aktivity s aplikací GalaxyCoder

V první části se žáci seznámí s aplikací GalaxyCoder. Jedná se o blokové vizuální programování, pro které žáci nepotřebují žádné nebo jen minimální znalosti programování. Uvnitř aplikace je tvorba algoritmů řešena intuitivně. Aby dosáhli cíle hry, musí najít správnou posloupnost jednotlivých bloků - příkazů. Prostřednictvím jednotlivých bloků dávají dohromady celý algoritmus a tím si sami rozvíjejí algoritmické myšlení.

Nástup jednotlivých bloků je postupný. Nejdříve se žáci seznamují s těmi nejjednoduššími algoritmickými kostrukty, teprve v dalších úrovních se postupně dostávají ke složitějším. Až do úrovně 5 používají hráči pouze pokyny: *jdi dopředu* nebo *otoč se doleva/doprava*. Posléze přibývá pokyn přeskoč. Od 11. úrovně je možné využít *cyklus*, od úrovně 27 je možné použít příkaz *čekej*.

Prostředí aplikace je atraktivní a zároveň i motivační. Součástí aplikace je celý příběh a zároveň jsou za jednotlivé úrovně k dosažení motivační kartičky, hvězdičky a další odměny.

Žákům se v aplikaci GalaxyCoder dostává okamžité zpětné vazby prostřednictvím ani-

mace postavy mimozemšťana. Pokud zvolí nesprávné řešení, hned to pochopí. Pokud jdou správným směrem, jsou žáci ihned odměněni.

Využití této aplikace je plánováno pro první dvě šedesáti minutové lekce. Zvláštností je, že je aplikace pouze v angličtině a ve slovenštině, ale předpokladem je, že by to nemuselo vadit.

Tabulka 5.1: Algoritmické konstrukty aplikace GalaxyCodr

Úroveň	Obecné algoritmické konstrukty	Algoritmické konstrukty aplikace
1-5	**posloupnost**	**jdi dopředu o [číslo] kroků** **otoč se doleva** **otoč se doprava**
6-10	posloupnost	jdi dopředu o [číslo] kroků otoč se doleva otoč se doprava **přeskoč**
11-21	posloupnost **cyklus s počtem opakování**	jdi do předu o [číslo] kroků otoč se doleva otoč se doprava přeskoč
22-26	posloupnost	jdi dopředu o [číslo] kroků otoč se doleva otoč se doprava
27-30	posloupnost	jdi dopředu o [číslo] kroků otoč se doleva otoč se doprava **čekej**
31-42	posloupnost	jdi dopředu o [číslo] kroků otoč se doleva otoč se doprava čekej přeskoč

V tabulce 5.1 jsou uvedeny na sebe navazující algoritmické konstrukty této aplikace. Zvýrazněné konstrukty jsou pro žáky v daném okamžiku nové. Aplikace využívá nejjednodušší konstrukt *posloupnost*. Celá aplikace využívá jen příkazy: *jdi dopředu, otoč se, přeskoč* a *čekej*. Jedná se o ty nejjednodušší pokyny, které žáci sami bez pomoci dokáží použít.

5.2 Aktivity s aplikací LightBot

Další výuková lekce je věnována aplikaci Lighbot. Princip hry je velmi podobný jako u Galaxy Codr. Celá aplikace má 3 lekce, celkem 20 úrovní. Postup využití jednotlivých konstruktů je však mnohem rychlejší. V první úrovni se využívá pouze *pohyb vpřed* a *rozsviť*. Ve druhé úrovni přibývá *otoč se doleva/doprava*. Ve třetí úrovni je navíc pohyb nahoru/dolu - *skoč*. Od druhé lekce přibývají procedury, které jsou pro žáky poměrně abstraktní. Jsou tu navíc jednotlivé procedury. V případě této aplikace jsou to stále se opakující procedury. Pokud bude více času, pak je možné žákům vysvětlit postup.

Vzhledem k tomu, že této aplikaci byla věnována pouze jedna hodina, vyšlo to tak, že žáci absolvovali právě 1. lekci, tedy 8 úrovní.

Další změnou je, že se postava pohybuje ve dvou rovinách (navíc i skáče). Hra nemá příběh. Cílem postavy je vždy jen dojít na určité políčko a provést zadanou aktivitu (rozsvítit žárovku).

Aplikace se ještě liší tím, že existuje více řešení, kterými mohou žáci dojít ke stejnému cíli. Přestože je tato hra v českém jazyce, není to zde vůbec důležité. Jednotlivé kroky, ze kterých si hráč vybírá, jsou namalované. Také zde nenajdeme klasické odměny za splněný úkol. Odměnou je rozsvícení žárovky a tím splní úkol.

Žáci si opět rozvíjí algoritmické myšlení prostřednictvím sestavování bloků kódů. Aplikace má vynikajícího obrázkového průvodce. Jediný pokyn, který stačí žákům dát, je: „Doveď robota na modré pole a rozsviť žárovku“. Zbytek obstará sám průvodce. Znamená to, že každý žák může jít vlastním tempem.

V tabulce 5.2 jsou uvedeny na sebe navazující algoritmické konstrukty této aplikace. Zvýrazněné konstrukty jsou pro žáky nové. Stále je řešen jen nejjednodušší konstrukt posloupnost. Příkazy byly obohaceny o pokyn *rozsviť žárovku* a *vyskoč/seskoč*.

Tabulka 5.2: Algoritmické konstrukty aplikace Lightbot

Úroveň	Obecné algoritmické konstrukty	Algoritmické konstrukty aplikace
1	**posloupnost**	**jdi dopředu** **rozsviť žárovku**
2	posloupnost	jdi dopředu **otoč se doleva** **otoč se doprava** rozsviť žárovku
3-8	posloupnost	jdi dopředu otoč se doleva otoč se doprava rozsviť žárovku **vyskoč nahoru / seskoč dolů**

5.3 Aktivity s aplikací BlocklyGames

Jedná se o aplikaci od firmy Google. Nachází se zde 8 edukačních her. Všechny hry jsou zaměřeny pro výuku programování. Bohužel pro využití na 1. stupni základní školy se nehodí všechny hry. Hned ve třetí hře je směr nastavování pomocí úhlu. Žáci 4. a 5. třídy ještě pojem velikost úhlu dobře neznají. Dokáží poznat trojúhelník a čtyřúhelník, vědí co je to pravý úhel, ale spolehnout se na to, že dokáží pracovat s velikostí úhlu nelze.

Pro naše účely se jeví jako nejlepší hra s názvem Bludiště. Žáci skládají jednotlivé bloky příkazů tak, aby se postava dostala do cíle znázorněného tradičním googlovským špendlíkem. Nic dalšího navíc postava nedělá, jen se přemísťuje. Celkem je zde 10 úrovní. Od první úrovně se využívají příkazy *pohyb vpřed, otočit doleva/doprava*. Od úrovně 3 přibývá *cyklus*, od úrovně 6 také *podmínka (if)* a od úrovně 9 je možné využít *podmínku (if - else)*. Zajímavé v této aplikaci je, že od úrovně 3 je omezen použitých bloků. Hra tak nutí žáky přemýšlet o nejvhodnějším řešení. Tvary jednotlivých bloků mají takovou podobu, že je nelze spojit do chybného pokynu.

Bludiště nemá děj, jedná se spíše o minihry. Po každé úspěšně splněné úrovni aplikace ukáže ekvivalent toho, co žák naprogramoval v jazyku JavaScript. Velkou výhodu může aplikace mít při výuce tohoto programovacího jazyka.

Algoritmické konstrukty této aplikace jsou uvedeny v tabulce 5.3. Kromě nejjednoduš-

šího konstruktu posloupnost je zde využito i dalších možností jako je *nekonečný cyklus* a *podmínka*. Novinkou je podmínka ve dvou variantách. Žáci postupují od jednodušší *podmínky* (pokud) ke složitější (pokud jinak), která se objevuje až ve dvou nejvyšších úrovních.

Tabulka 5.3: Algoritmické konstrukty aplikace BlocklyGames

Úroveň	Obecné algoritmické konstrukty	Algoritmické konstrukty aplikace
1-2	**posloupnost**	**pohyb vpřed** **otočit levá** **otočit levá**
3-5	posloupnost **nekonečný cyklus**	pohyb vpřed otočit levá otočit pravá
6-8	posloupnost nekonečný cyklus **podmínka (pokud)**	pohyb vpřed otočit levá otočit pravá
9-10	posloupnost nekonečný cyklus podmínka (pokud) **podmínka (pokud - jinak)**	pohyb vpřed otočit levá otočit pravá

5.4 Aktivity s programem Scratch

V 5. lekci se žáci dostanou k dětskému programovacímu jazyku. Zatímco v předchozích lekcích byly jasně zadané úkoly, tady jsou úkoly zadávané již vyučujícím. Žáci se v tomto programovacím prostředí učí chápat reálné algoritmické koncepty. Příkazy jsou tu opět ve formě barevných bloků.

Přestože lze nalézt na webových stránkách Scratch mnoho hotových a připravených výukových materiálů, byla vytvořena vlastní zadání pro účely programovacího kroužku. Přestože vypadá tento blokový programovací jazyk velmi jednoduše, lze jeho prostřednictvím naprogramovat i komplexní program.

Algoritmické konstrukty používané v dětském programovacích jazyku Scratch jsou uvedeny v tabulce 5.4. Konstrukty, které jsou pro žáky nové jsou zvýrazněny.

Tabulka 5.4: Algoritmické konstrukty ve Scratch

Lekce	Obecné algoritmické konstrukty	Algoritmické konstrukty jazyka Scratch
5 - Základy		
6 - Kreslení	**posloupnost**	**dopředu o [číslo] kroků** **nastav směr [číslo]**
7 - Autodráha	posloupnost **spuštění události**	dopředu o [číslo] kroků **nastav barvu pera** nastav směr [číslo] **nastav tloušťku pera** **po stisku klávesy [klávesa]** **pero zapni** **smaž**

8 - Poušť a souřadnice	**čekej dokud nenastane [pod.]** posloupnost spuštění události	**dotýká se [objekt]** **skryj se** **ukaž se** **když narazíš na okraj odraz se** **otiskni se** klouzej [číslo] sekund na [souřadnice X],[souřadnice Y] po stisku klávesy [klávesa] pero zapni
9 - Plošinky	**nekonečný cyklus** čekej dokud nenastane [podmínka]	**bublina [text]** **skoč na x: [číslo] y:[číslo]** nastav směr [číslo] dopředu o [číslo] kroků zastavit vše když narazíš na okraj odraz se ukaž se

10 - Kreslení a Breakout	**podmínka (if)** nekonečný cyklus čekej dokud nenastane [podmínka]	**nastav X na [X myši]** **nastav Y na [Y myši]** **dotýkáš se barvy [barva]** scénář nastav směr když narazíš na okraj, odraz se nastav barvu pera pero zapni/vypni zastav všechno
11 - Pong - hra pro dva	**proměnná** **nastav [proměnná] na [číslo]** **změň [proměnná] o [číslo]** **náhodné číslo od [číslo] do [číslo]** nekonečný cyklus podmínka (if)	**změn y o [číslo]** **otoč se [leva/pravá] o [číslo] stupňů** **klávesa [klávesa] stisknuta?** když narazíš na okraj, odraz se dotýkáš se barvy [barva]

12 - Body	nastav [proměnná] na [číslo] změň [proměnná] o [číslo] nekonečný cyklus čekej dokud nenastane	**nastav směr k [objekt]** **bublina [text], [číslo] sekund** skryj se dotýkáš se zastav všechno
13 - Stopky	**čekej [číslo] sekund** nekončený cyklus nastav [proměnná] na [číslo] změň [proměnná] o [číslo] podmínka (if)	**po kliknutí na mě**
14 - Klikání na balóny	**načtení uživatelského vstupu** náhodné číslo od [číslo] do [číslo] nastav [proměnná] na [číslo] změň [proměnná] o [číslo] podmínka (if) čekej [číslo] sekund	**nastav velikost na [číslo] %** **otázka** **změň velikost o [číslo]** **přepni pozadí** **změň velikost o [číslo]** **skoč na x: [číslo] y: [číslo]** **odpověď**

15 - Tipovací hra	**porovnávání proměnných** náhodné číslo od [číslo] do [číslo] nastav [proměnná] na [číslo] změň [proměnná] o [číslo] podmínka (if - else) čekej [číslo] sekund načtení uživatelského vstupu spojení řetězců textu	bublina [text], [číslo] sekund odpověď
16 - Závěrečná hra	událost nastav [proměnná] na [číslo] změň [proměnná] o [číslo] podmínka (if - else) čekej [číslo] sekund načtení uživatelského vstupu nekonečný cyklus čekej dokud nenastane [podmínka] čekej [číslo] sekund	**změň kostým** otázka po stisku klávesy skryj se ukaž se odpověď dotýkáš se barvy nastav směr [číslo]

Část II

Experimentální část

6 Výzkumný projekt

V experimentální části jsou ověřovány vybrané prostředky rozvoje algoritmického myšlení ve směru naplnění cíle $\mathbf{C_4}$. Znamená to realizovat a ověřit výuku vybraných programů pro zlepšení rozvoje algoritmického myšlení. Ověřování navržených metodických postupů proběhlo v prostředí programovacího kroužku na základní škole. Akční výzkum byl použit pro ověření funkčnosti vybraných nástrojů.

6.1 Charakteristika programovacího kroužku na základní škole

Akční výzkum proběhl v rámci kroužku informatiky na základní škole v Praze. Škola byla vybrána na základě deklarované podpory digitálních technologií. Jedná se o velkou školu s více než 700 žáky na prvním i druhém stupni.

Kroužek programování pro první stupeň probíhá každou středu od 14 do 15 hodin v počítačové učebně. Na škole aktuálně probíhají dva kroužky programování ve stejnou chvíli. Akční výzkum byl aplikován v kroužku začátečníků. Jedná se o žáky 4. a 5. třídy, kteří do tohoto kroužku začali chodit poprvé od začátku školního roku 2018/19. Akčního výzkumu se zúčastnilo všech 8 žáků. Výzkum byl realizován ve dvanácti 60 minutových hodinách v průběhu prvního pololetí.

Rozdělení žáků do dvou částí se jevilo jako neetické, protože část žáků by nedostala adekvátní příležitost pro osvojení si algoritmického myšlení.

6.2 Aplikace akčního výzkumu

Akční výzkum v oblasti vzdělávání má za cíl zvýšit profesionalitu učitele. Těžiště tohoto typu výzkumu leží v pedagogické praxi. Prostřednictvím opakovaného zavádění změn

umožňuje zlepšování kvality poskytovaného vzdělávání. To jsou důvody, proč byl akční výzkum vybrán pro aplikaci v této práci. Tento výzkum je praktický a je založen na subjektivních názorech autora.[1] Studuje reálnou školní situaci.

Tato práce vychází z toho, že akční výzkum je podle Stephena Kemmise[2] sebereflexe pedagogické situace, která vede ke zkvalitnění pedagogické praxe. Akční výzkum této práce vycházel z otázky: Jak zkvalitnit pedagogickou praxi při rozvoji algoritmického myšlení u žáků ve věku 10 až 11 let?

V rámci tohoto výzkumného projektu byly shromažďovány informace. V každé lekci proběhlo pozorování a rozhovory se žáky. Poté došlo k vyhodnocení výsledků. Na základě získaných zkušeností z předchozích lekcí byly připravovány a upravovány následující výukové hodiny.

6.3 Popis výukových lekcí

V této kapitole jsou podrobně rozebrány jednotlivé lekce. Jedná se o návod použitelný i dalšími vyučujícími. V úvodu jsou vždy uvedeny cíle, potřebné předběžné znalosti (prekoncept) a podrobný postup. V poznámkách na konci každé lekce jsou uvedeny užitečné informace pro vyučujícího. Všechny zkušenosti z probíhající výuky jednotlivých lekcí jsou uvedeny až v kapitole 7.

6.3.1 Lekce 1 - Zkoumání aplikace GalaxyCodr

Lekce se věnuje plnění úrovní v aplikaci GalaxyCodr. Žáci se prvně seznamují se základními algoritmickými konstrukty.

[1] Robert Chambers. *Practical action*. 2. vyd. Corwin Press, 2018, s. 190–218. isbn: 1483362132. doi: 10.4324/9781315835815-8.

[2] Danuše Nezvalová. Akční výzkum ve škole. In: *Pedagogika* 53 (2003), s. 300–308. Dostupné z: pages.pedf.cuni.cz/pedagogika/?attachment_id=1944&edmc=1944%0A%0A.

Cíle výuky

- Žák se seznámí s hrou GalaxyCodr;
- žák pochopí způsoby pohybu postavy na herní ploše;
- žák si uvědomí použití algoritmických konstruktů: *jdi do dopředu, otoč se doprava / doleva, přeskoč.*

Prekoncept

- Žák využije vlastní znalosti získané například z hraní počítačových her.

Postup

- Na začátku hodiny učitel požádá žáky, aby si otevřeli webový prohlížeč;
- dále je instruuje přejít na adresu *galaxycodr.com/sk*;
- učitel vysvětlí žákům, že si v pravého horním rohu mohou vybrat jazyk, ve kterém pak bude aplikace fungovat;
    - při volbě jazyka je pro žáky výběr mezi slovenštinou a angličtinou (vzhledem k minimu pokynů je předpoklad, že s jazyky nebude žádný problém);
- vybrat a kliknout na variantu „Hraj zdarma“;
- učitel zaregistruje žáky, tak aby jim dosažená úroveň zůstala i do další lekce;
- instruktážní videa žákům vysvětlí fungování aplikace.

Poznámky

- Drobnou nevýhodou této aplikace je absence varianty v českém jazyce, vyučující je připraven v případě problému jednoduché příkazy přeložit;
- vzhledem k tomu, že v následující lekci bude navazovat další částí této aplikace, je vhodné informovat nepřítomné žáky;
- je možné žáky upozornit na závislost mezi správným řešením úrovně a množství získaných odměn.

Obrázek 1: Ukázka úrovně 14 aplikace GalaxyCodr

6.3.2 Lekce 2 - Dokončení všech úrovní aplikace GalaxyCodr

V této lekci žáci dokončí většinu úrovní v aplikaci GalaxyCodr. Rychlejší žáci se mohou věnovat získávání odměn za správně vyřešené úrovně.

Cíle výuky

- Žák dokončí všechny úrovně aplikace GalaxyCodr;
- žák si zopakuje základní povely pro pohyb postav;
- žák si osvojí nový povel *přeskoč*;
- žák si procvičuje prostorovou představivost;
- žák přidává další algoritmický konstrukt *čekej* (obrázek 2).

Prekoncept

- Žák využívá znalosti získané z předchozí lekce.

Obrázek 2: Ukázka druhé části aplikace GalaxyCodr

Postup

- Na začátku hodiny učitel zopakuje postup otevření a přihlášení se do aplikace;
- v případě potřeby podá učitel instrukce žákům, kteří nebyli přítomni na minulé hodině;
- učitel reaguje na dotazy žáků, jinak nechá žáky volně pracovat.

Poznámky

- Druhá část této aplikace má větší nároky na prostorovou představivost.

6.3.3 Lekce 3 - Zkoumání aplikace LightBot

Lekce je zaměřena na řešení úkolů v aplikaci Lightbot. Oproti předchozí aplikaci GalaxyCodr, zde přibývá i činnost (*rozsviť žárovku*).

Cíle výuky

- Žák se seznámí s hrou LightBot;

- žák se orientuje v novém prostředí a chápe pokyny k pohybu postavy;
- žák přidává další algoritmický konstrukt rozsviť;
- žák chápe pohyb postavy ve více rovinách;
- žák objevuje existenci více možností k dosažení cíle.

Prekoncept

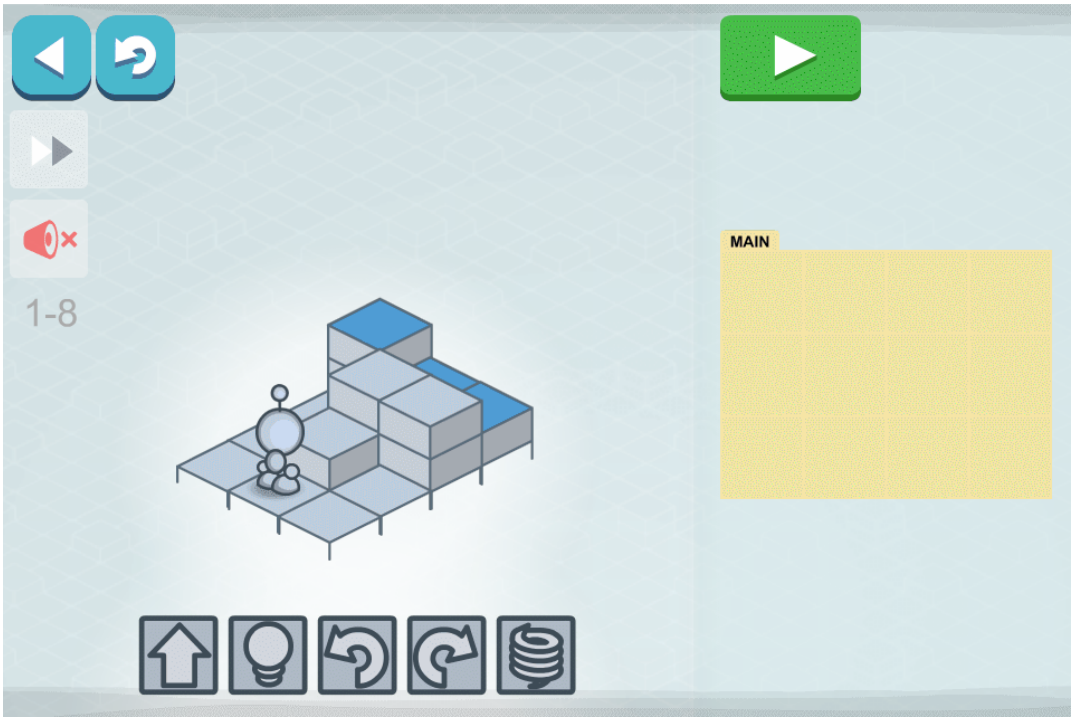
- Žák využívá znalostí získaných ve hře GalaxyCodr.

Postup

- Učitel požádá žáky o otevření webového prohlížeče kompatibilního s aplikací např. ve Firefoxu ve verzi 78.0.1;
- dalším pokynem je otevření strany stránky *lightbot.com* a povolení spuštění aplikace v programovacím jazyku Flash;
- učitel vyzve k výběru jazyka v pravém horním rohu (angličtina, čeština);
- v levém horním rohu si žáci vyberou barvu robota;
- následuje pokyn pro výběr první úrovně z první kategorie;
- automaticky dojde ke spuštění animace k vysvětlení fungování aplikace.

Poznámky

- Vyhovující algoritmické konstrukty jsou pouze v první kategorii;
- pro účely této práce se nepředpokládá postup do vyšších kategorií;
- na rozdíl od předcházející aplikace se ve hře LightBot objevují nové konstrukty výrazně rychleji;
- v případě potřeby je možné informovat žáky o omezení v podobě počtu použitých příkazů.



Obrázek 3: Ukázka úrovně 8 aplikace LightBot

6.3.4 Lekce 4 - Zkoumání aplikace Blockly Games

Poslední lekce s prací v aplikaci pracuje v programu Blocky games. Oproti třem předchozím lekcím zde žáci poprvé pracují s konstruktem *podmínka*.

Cíle výuky

- Žák se seznámí s aplikací Blockly games;
- žák se orientuje v novém prostředí a chápe pokyny k pohybu postavy;
- žák přidává k předchozím další algoritmické konstrukty: *nekonečný cyklus, podmínku if a if - else*.

Prekoncept

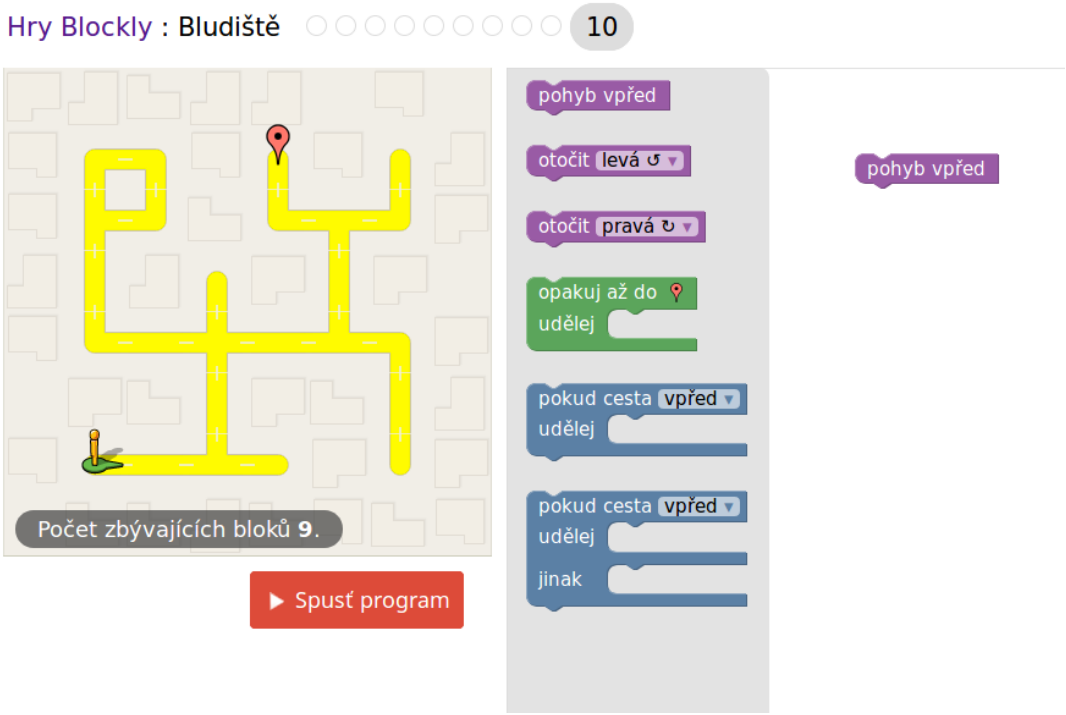
- Žák využívá znalostí z předcházejících hodin.

Postup

- Učitel vydá pokyn k otevření webového prohlížeče;
- např. prostřednictvím tabule žáky seznámí s adresou *https://blocklygames.com*;
- učitel upozorní na umístění ikonky s výběrem jazyka;
- učitel vyzve skupinu žáků, aby si všichni vybrali kategorii s názvem Bludiště;
- pokud je třeba, učitel může instruovat žáky o průběhu hry, protože se v aplikaci nenachází ukázková animace.

Poznámky

- Kategorie Bludiště byla vybrána jako nejvhodnější z důvodu navazujících algoritmických konstruktů (*if* a *if - else*);
- další kategorie nejsou pro žáky v tomto věku vhodné, protože vyžadující práci s velikostí úhlu.



Obrázek 4: Ukázka úrovně 10 aplikace Blockly games - bludiště

6.3.5 Lekce 5 - Zkoumání programu Scratch

Hlavním cílem lekce je seznámit žáky s prostředím programu Scratch. Žáci by měli pochopit základní principy a funkčnost jednotlivých části programu: příkazy, scéna, programovací oblast. Učitel předvádí jednotlivé koncepty a funkce programu Scratch a následně nechává žáky volně tvořit.

Cíle výuky

- Žák se seznámí s programem Scratch;
- žák se orientuje v jednotlivých pokynech pro pohyb a otáčení postavou a také pro kreslení postavou;
- žák chápe strukturu a vlastnosti jednoduchých scénářů;
- žák dokáže vytvořit scénář pro kreslení schodové struktury.

Prekoncept

- Žák využívá znalostí z blokového programování v předcházejících aplikací.

Postup

- Učitel vybízí žáky ke zkoumání základních bloků v programu Scratch;
- nabádá je, aby sami přišli na využití a případné spojení jednotlivých bloků;
- učitel se zajímá, zda žáci zjistili, jak pohybovat a kreslit s postavou;
- v případě potřeby je možné žákům předvést, jak má pohyb postavy vypadat. Pro lepší názornost je třeba po každé části pohybu příkaz *čekej*;
- v druhé části hodiny učitel zadává úkol naprogramovat kreslení schodové struktury.

Poznámky

- Je dobré pro učitele být připraven na zvídavé otázky při představování programu, kde není předem zadaný hlavolam v podobě hledání cesty.

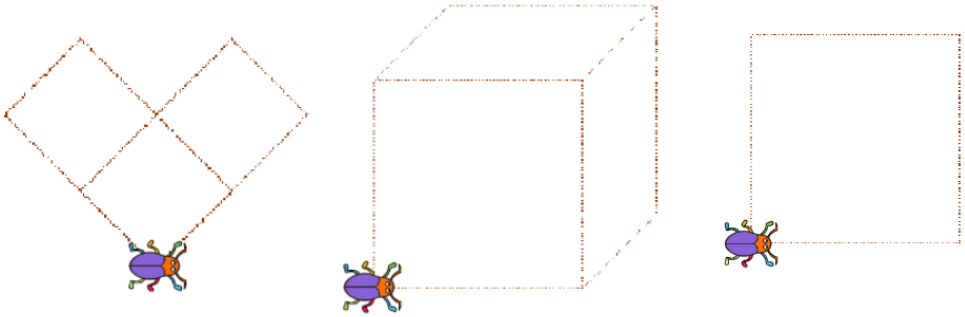
- Do určité míry je třeba obrnit se trpělivostí, je těžké upoutat pozornost žáků v okamžiku, kdy objeví např. nový zajímavý pohyb postavy.

6.3.6 Lekce 6 - Kreslení s programem Scratch

Tato lekce je zaměřena na kreslení a pohyb postav. V první části žáci kreslí list pily a to ve dvou polohách. Ve druhé pak žáci obkreslují různé tvary, jak je vidět na obrázku 5.

Jak je vidět v tabulce 6.1, lekce obsahuje celkově 5 vstupních souborů:

- Kreslení listu pily pod úhly 90° a 0°.[3]
- Kreslení listu pily pod úhly 45° a 135°.[4]
- Obkreslování čtverce.
- Obkreslování obrazce krychle.
- Obkreslování vzoru „srdce“.



Obrázek 5: Scény pro lekci 6

Cíle výuky

- Žák chápe systém seřazení několika bloků za sebou tak, aby vznikl funkční kód;

[3] Jiří Vaníček et al. *Programování ve Scratch pro 2. stupeň základní školy*. České Buďějovice: Jihočeská univerzita v Českých Buďějovicích, Pedagogická fakulta, 2020, s. 17. ISBN: 978-80-7394-783-5. Dostupné z: `https://imysleni.cz/ucebnice/programovani-ve-scratchi-pro-2-stupen-zakladni-skoly`, s. 6.

[4] Ibid.

- žák umí použít příkaz *pero zapnout*;
- žák dokáže použít velikost úhlů 45° a 90°;
- žák umí použít příkazy *dopředu o [číslo] kroků*, *nastav směr [číslo]*;
- žák dokáže sestavit blok příkazů tak, aby postava s perem dokázala nakreslit připravený vytečkovaný objekt;
- žák dokáže vytvořený program uložit do souborů;
- žák k předchozím algoritmickým konstruktům přidává pojem *sekvence* příkazů.

Prekoncept

- Žák využívá hlavně znalosti z úvodní hodiny programování v jazyce Scratch.

Postup

- Učitel seznamuje žáky s příkazem: po kliknutí na zelenou vlajku;
- učitel si ověřuje, že žáci znají úhel 45° a 90°;
- učitel vysvětluje, jak vytvořený program uložit do souborů;
- učitel zadává žákům vytečkované obrazce a vybízí k jejich obkreslování;
- učitel představuje žákům blok příkazu pro možnost resetování do výchozího stavu.

Tabulka 6.1: Soubory pro lekci 6 - Kreslení s programem Scratch

Název souboru	Odkaz
Lekce 6A - pila 1	`https://scratch.mit.edu/projects/511079929/`
Lekce 6B - pila 2	`https://scratch.mit.edu/projects/51107990/`
Lekce 6C - Kreslení (krychle)	`https://scratch.mit.edu/projects/504604406/`
Lekce 6D - Kreslení (srdce)	`https://scratch.mit.edu/projects/504603117/`
Lekce 6E - Kreslení (čtverec)	`https://scratch.mit.edu/projects/517749595/`

Poznámky

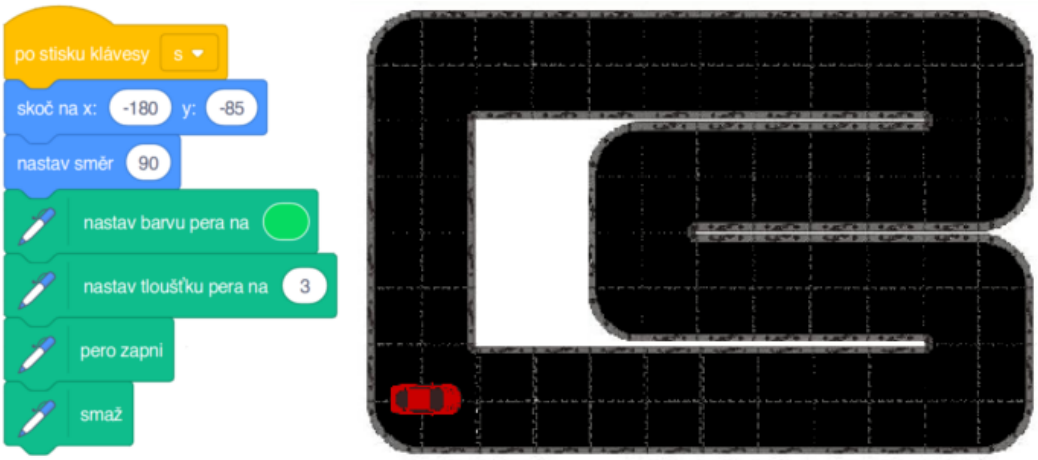
- Obrazce určené k obkreslení byly vybrány tak, aby obsahovaly pouze úhly 45° a 90°;
- žáci občas rozebírají blok kódu určený pro resetování do výchozího stavu.

6.3.7 Lekce 7 - Autodráha v programu Scratch

Vstupní soubor (tabulka 6.2) obsahuje připravenou scénu autodráhy, postavu v podobě auta a fragmenty kódu (viz obrázek 6). Hlavní myšlenkou lekce 7 je předvést žákům rozdíl mezi ovládáním postavy ručně např. pomocí šipek a automaticky pomocí připravených kroků.

Cíle výuky

- Žák umí vysvětlit příkaz po stisku klávesy;
- žák zjistí jakým způsobem lze měnit směr natočení postavy pomocí stisku klávesy;
- žák dokáže naprogramovat interaktivní ovládání autíčka na autodráze pomocí směrových kláves;
- žák umí naprogramovat autíčko tak, aby po dráze jezdilo samo.



Obrázek 6: Scéna pro lekci 7

Prekoncept

- Žák využívá znalostí z předchozích hodin programování v jazyku Scratch, hlavně kreslení rovných a lomených čar.

Postup

- Učitel otevírá žákům scénu s dráhou a postavou autíčka (obrázek 6);
- učitel vysvětluje připravený blok příkazů pro resetování scény, kterou mohou žáci využít pro navrácení autíčka na start, smazání všech čar a opětovné nastavení pera (obrázek 6);
- učitel promítne výsledný pohyb autíčka po dráze tak, aby žáci snadno pochopili zadání;
- učitel představuje žákům příkaz: *po stisku klávesy [klávesa]* a jeho využití;
- učitel vydává pokyn pro splnění prvního a posléze i druhé části úkolu.

Poznámky

- Pro nejrychlejší žáky je možné rozšířit zadání o kombinaci prvního a druhého úkolu přidáním dalšího autíčka, které žák ovládá a soutěží tak s autíčkem ovládaným programem.

Tabulka 6.2: Soubory pro lekci 7 - Autodráha v programu Scratch

Název souboru	Odkaz
Lekce 7 - Autodráha [zadání]	`https://scratch.mit.edu/projects/501422425`
Lekce 7 - Autodráha [řešení]	`https://scratch.mit.edu/projects/517771437`

6.3.8 Lekce 8 - Poušť a souřadnice v programu Scratch

Tato lekce je rozdělená na dvě části. V první části se využívá znalostí z předchozí lekce a kombinuje ovládání postavy hráčem společně s automatickým ovládáním postavy.

Jedná se tak o první „hru“. Hráč pohybuje postavou a snaží se „sníst“ postavy jídla. Na scéně se pohybuje také postava myši, které se hráč musí vyhnout, jak je vidět na obrázku 7.

Druhá část lekce je zaměřena na vysvětlení systémů kartézských souřadnic a jejich využití v programu Scratch. Scéna je vidět na obrázku 8. Žáci kreslí obrazce definované učitelem pomocí příkazu *klouzej [číslo] sekund na souřadnice [číslo],[číslo].*

Cíle výuky

- Žák se seznamuje s ovládáním více různých postav;
- žák vytváří funkční model scény, kde se pohybuje více postav;
- žák se seznamuje s osami x a y;
- žák procvičuje nákres obrazců trojúhelníku, čtverce a domečku;
- žák získává další algoritmické konstrukty: *dotýká se, ukaž se, když narazíš na okraj, odraz se*;
- žák zkoumá vytváření algoritmů pro vzájemnou interakci více postav;
- žák dokáže vysvětlit spojitost postavy s příslušným algoritmem;
- žák dokáže využít příkaz *klouzej [číslo] sekund na souřadnicích [číslo],[číslo]*;
- žák chápe výhody používání souřadnicové sítě.

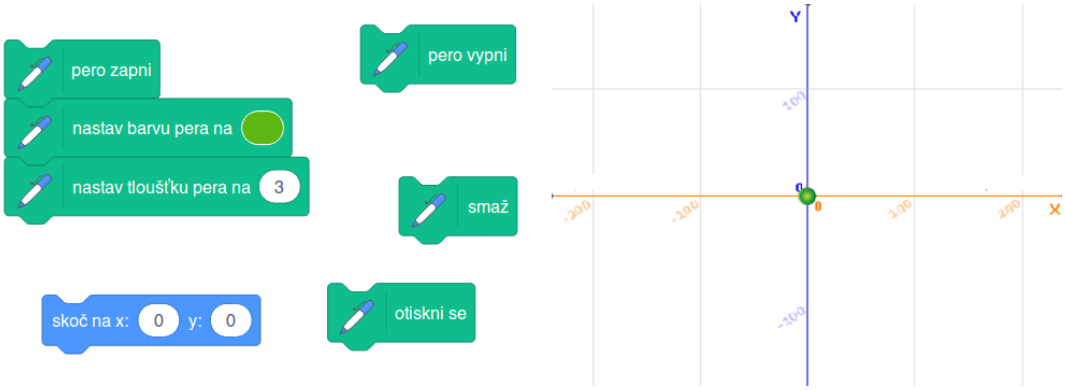
Prekoncept

- Žák využívá znalostí z předcházejících hodin.

Postup

- Učitel představuje pozadí první scény a předvede postavu brouka pro pohyb, postavu myši pro lovení a vhodné postavy k jídlu (obrázek 7);
- učitel žáky seznamuje s novými příkazy či spojitostí postavy s příslušným algoritmem;
- učitel vyzve žáky k vytvoření prvního úkolu;

Obrázek 7: První scéna pro lekci 8



Obrázek 8: Scéna a část připraveného kódu pro druhou část lekce 8

- v druhé části hodiny učitel představuje pozadí se čtvercovou sítí a postavou s rozšiřujícími příkazy pro obsluhu pera (obrázek 8);
- učitel zadává druhý úkol - vytvoření obrazců pomocí pohybu po souřadnicové síti.

Poznámky

- Druhá část hodiny, která je věnována souřadnicové síti, je přípravou na další lekce.

Tabulka 6.3: Soubory pro lekci 8 - Poušť a souřadnice v programu Scratch

Název souboru	Odkaz
Lekce 8A - poušť [zadání]	`https://scratch.mit.edu/projects/511093604`
Lekce 8A - poušť [řešení]	`https://scratch.mit.edu/projects/517390007`
Lekce 8B - Souřadnice[zadání]	`https://scratch.mit.edu/projects/511093767`
Lekce 8B - Souřadnice [řešení]	`https://scratch.mit.edu/projects/517391384`

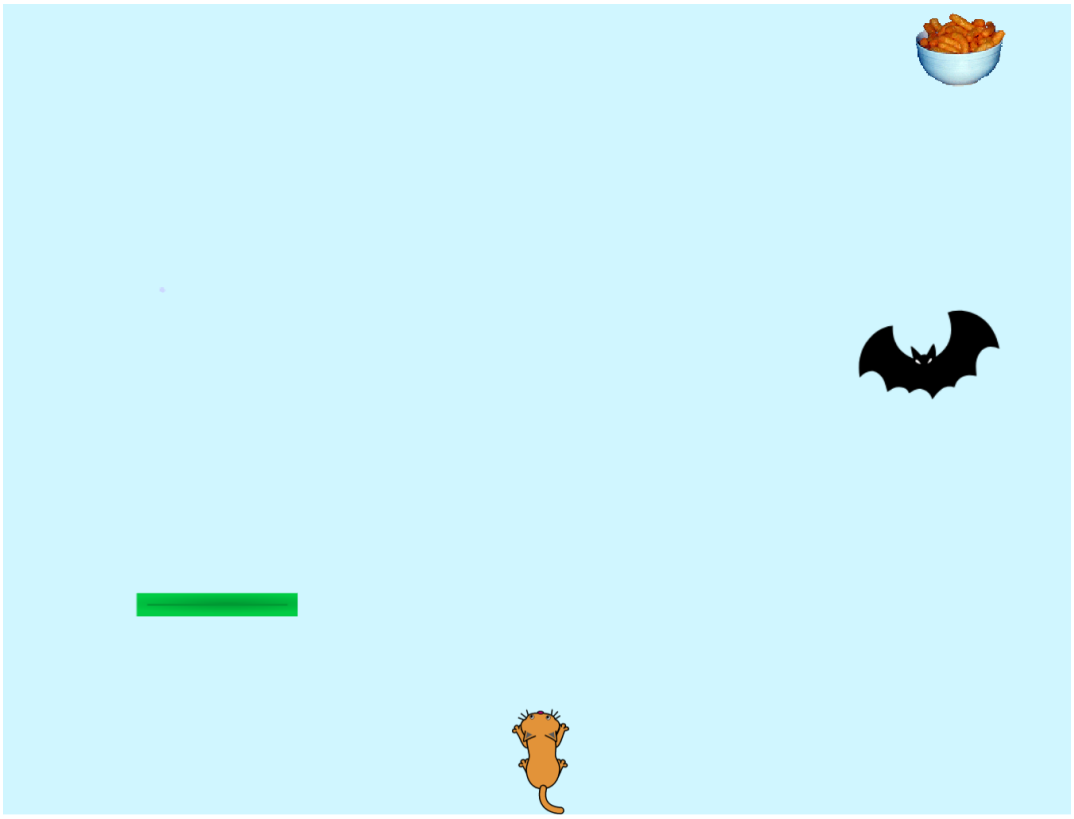
6.3.9 Lekce 9 - Plošinky s programem Scratch

Lekce Plošinky rozšiřuje předchozí lekci v podobě hry a přidává více postav. Dále se zaměřuje na komunikaci s uživatelem v podobně příkazu *bublina [text]*. Scéna pro tuto lekci je vidět na obrázku 9.

Cíle výuky

- Žák chápe, jakým způsobem vytvořit vlastní hru;
- žák dokáže naprogramovat postavu, kterou sám ovládá a navíc programuje automatický pohyb dalších postav;

- žák k předchozím algoritmickým konstruktům přidává příkazy bublina a zastavit vše.



Obrázek 9: Scéna lekce 9

Prekoncept

- Žák využívá předcházejících znalostí programu Scratch, především pohyb postav pomocí souřadnicové sítě;
- žák také využívá již získané znalosti příkazu: dotýká se.

Postup

- Učitel představí scénu hry a 3 postavy:
  - postava kočky, jejímž cílem bude dojít k misce s jídlem;
  - postavu netopýra, který se pohybuje náhodnými odrazy od stěn;
  - postavu plošinky, která bude naprogramována pouze na vodorovný pohyb.

- Učitel upozorní žáky, že po dotyku kočky s některou z postav, hra končí, pokud se dostane kočka k misce s jídlem, hráč vítězí;
- učitel vysvětlí důležitost vytváření algoritmu pro každou postavu.

Poznámky

- Pokud existují žáci, kteří chyběli předcházející hodinu, je výhodné s nimi zopakovat práci se souřadnicovou sítí;
- pokud jsou žáci, kteří stihnou naprogramovat hru rychle, pak je možné jim nabídnout rozšíření:
  - přidání dalších postav;
  - změna kostýmu kočky po dotyku například v na kostým ducha.

Tabulka 6.4: Soubory pro lekci 9 - Plošinky s programem Scratch

Název souboru	Odkaz
Lekce 9 - Plošinky [zadání]	`https://scratch.mit.edu/projects/504605035/`
Lekce 9 - Plošinky [řešení]	`https://scratch.mit.edu/projects/504604653/`

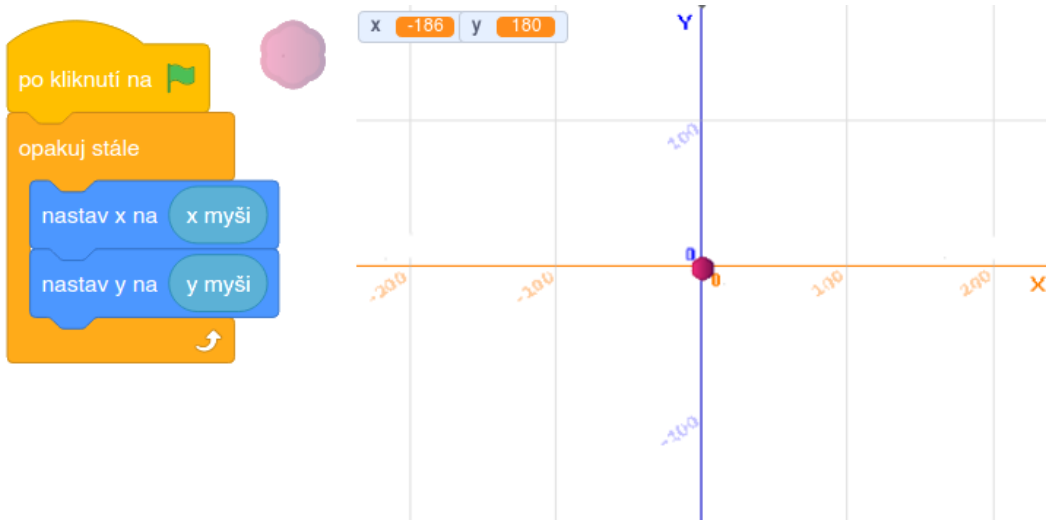
6.3.10 Lekce 10 - Kreslení a Breakout s programem Scratch

První část lekce se věnuje opakování práce se souřadnicemi (obrázek 10). Druhá část pokračuje s programováním her. Žáci si vyzkouší naprogramovat hru ve stylu Breakout (obrázek 11).

Cíle výuky

- Žák umí naprogramovat myš tak, aby s ní mohl kreslit po scéně;
- žák si upevňuje znalosti práce se souřadnicemi;
- žák dokáže vysvětlit využití souboru příkazů pro sledování pozice kurzoru myši - *x myši a y myši* s *nastav na x: [číslo] a nastav y: [číslo]*;

- žák umí naprogramovat hru podle zadání s poskytnutím vstupního souboru;
- k předcházejícím konstruktům je přidán další pokyn: *dotýkáš se barvy* a poprvé v tomto programu je použit konstrukt *podmínky*.



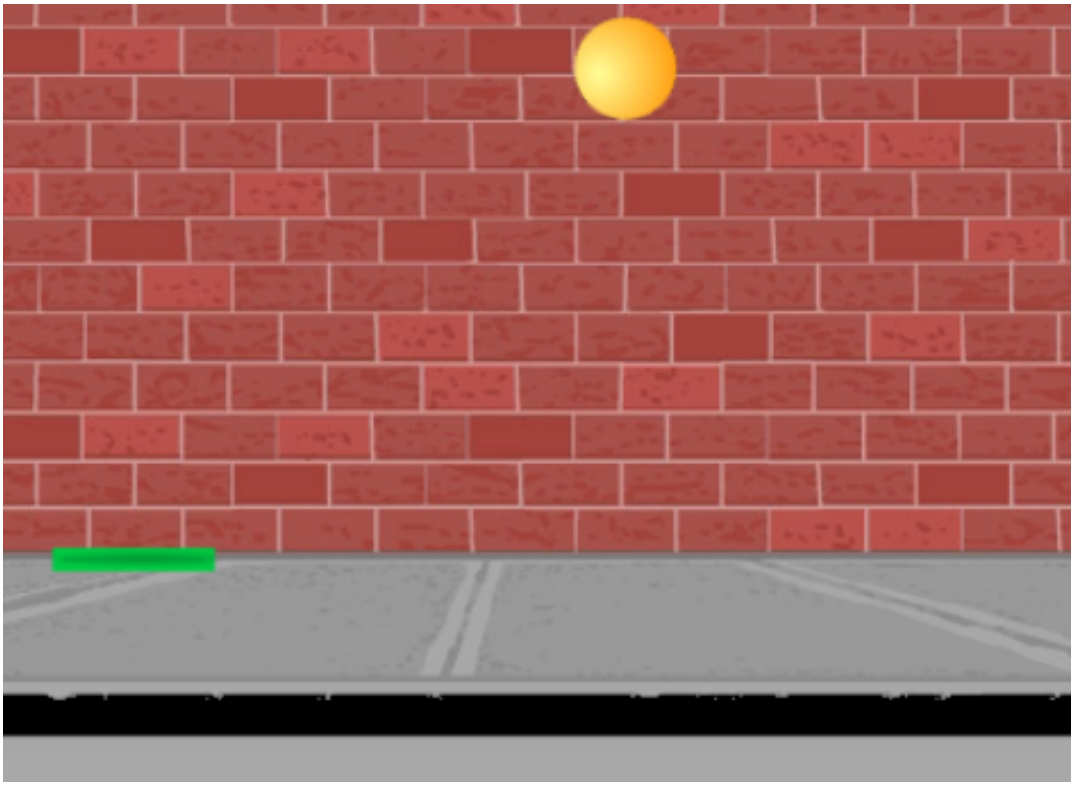
Obrázek 10: Scéna a část kódu první části lekce 10

Prekoncept

- Žák využívá hlavně znalosti a zkušenosti s prací se souřadnicemi z předchozích úloh.

Postup

- Učitel představuje výchozí scénu se souřadnicovou sítí pro první část hodiny;
- pro usnadnění je žákům představena část kódu, který zajistí pohyb postavy pomocí pohybu myší;
- ve druhé části hodiny učitel představuje hru se dvěma postavami:
  - postava míčku obsahuje připravený kus kódu pro výpočet odrazu;
  - postava plošinky má připravenou část kódu pro pohyb pomocí myši.
- Scéna představená učitelem obsahuje černou čáru, podle které lze poznat, že míček minul plošinku.



Obrázek 11: Scéna druhé části lekce 10

Poznámky

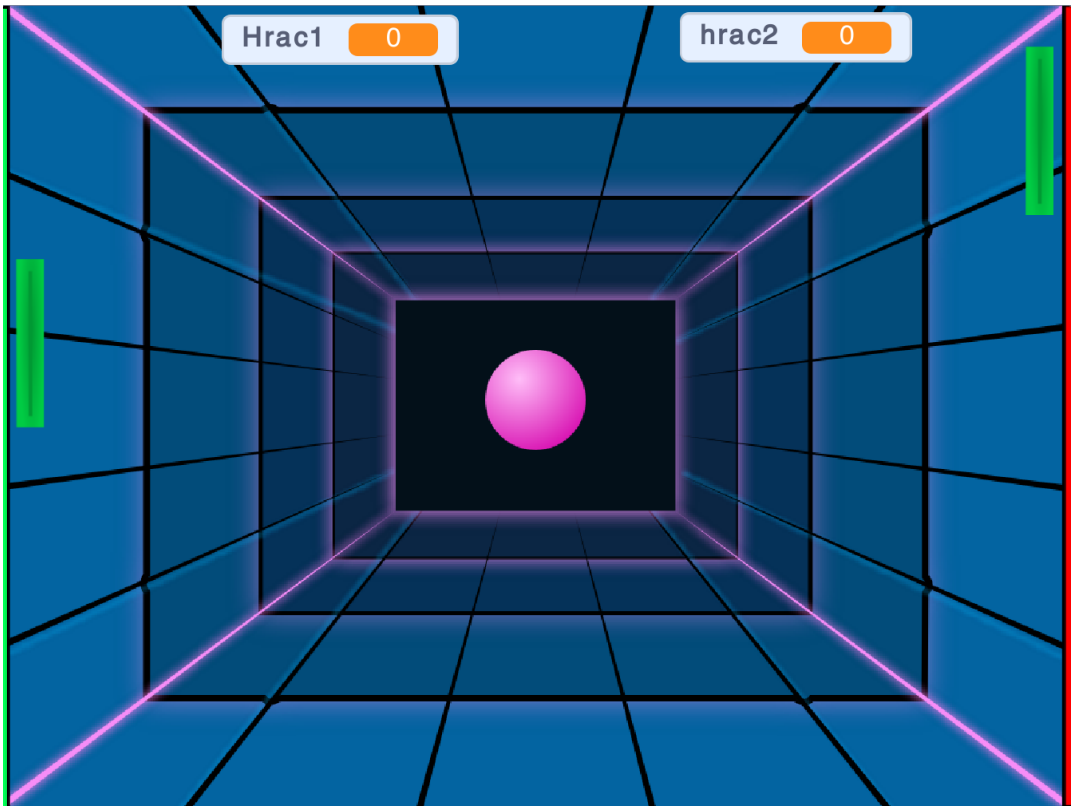
- Připravený kód ukazuje polohu ukazatele myši v souřadnicové síti;
- výpočet odrazu pro žáky je příliš složitý, proto je předem připravený hotový scénář, který odraz vypočítá jako 180 - aktuální směr ve stupních.

Tabulka 6.5: Soubory pro lekci 10 - Kreslení a Breakout s programem Scratch

Název souboru	Odkaz
Lekce 10A - Kreslení [zadání]	`https://scratch.mit.edu/projects/504603988/`
Lekce 10B - Breakout [zadání]	`https://scratch.mit.edu/projects/504610053/`
Lekce 10B - Breakout [řešení]	`https://scratch.mit.edu/projects/504653397/`

6.3.11 Lekce 11 - Pong - hra pro dva v programu Scratch

Lekce pracuje s prekoncepty z minulé lekce a rozšiřuje je o možnost interakce dvou uživatelů (hráčů). Žáci si vyzkouší programování klasické hry Pong. Scéna (obrázek 12) využívá stejných příkazu jako předchozí lekce: *dotýkáš se barvy [barva].*



Obrázek 12: Scéna lekce 11

Cíle výuky

- Žák dokáže propojit dosavadní znalosti pro vytvoření hry Pong;
- žák chápe koncept proměnné pro počítání skóre ve hře.

Prekoncept

- Žák využívá znalosti z programování hry Breakout.

Postup

- Učitel představuje výchozí scénu se třemi postavami;
- učitel zdůvodňuje barvy na okrajích scény určené pro počítání bodů;
- učitel vysvětluje koncept *proměnné*;
- učitel předává žákům vstupní soubor s fragmenty kódu pro začátek hry.

Poznámky

- Ze stejného důvodu jako v předchozí lekci je i zde připravená ta část kódu, kde dochází k výpočtu odrazu míčku od plošinky.
- Nadanějším žákům, je možné zadání ztížit v podobě práce s proměnnými - počítání skóre.

Tabulka 6.6: Soubory pro lekci 11 - Pong - hra pro dva v programu Scratch

Název souboru	Odkaz
Lekce 11 - Pong pro dva [zadání]	`https://scratch.mit.edu/projects/504610373/`
Lekce 11 - Pong pro dva [řešení]	`https://scratch.mit.edu/projects/504652873/`

6.3.12 Lekce 12 - Body - proměnné v programu Scratch

Hlavním cílem lekce 12 je rozšířit znalosti žáků v oblasti práce s proměnnými. Výuka tak navazuje na předchozí lekci, kde byla možnost rozšířit práci s proměnnými pro nadanější žáky.

Cíle výuky

- Žák chápe principy práce s proměnnými;
- žák si osvojí využití příkazů: *nastav směr k [objekt], bublina [text], [číslo] sekund*;

Prekoncept

- Žák využívá znalostí z předchozí hodin zejména v oblasti pohybu postav.

Postup

- Učitel představuje výchozí scénu se čtyřmi postavami (tabulka 6.7);
- učitel vysvětluje koncept *proměnné*;
- učitel vysvětlí jakým způsobem se budou nastavovat a zvyšovat hodnoty proměnných;
- učitel předává žákům vstupní soubor s fragmenty kódu pro začátek hry.

Poznámky

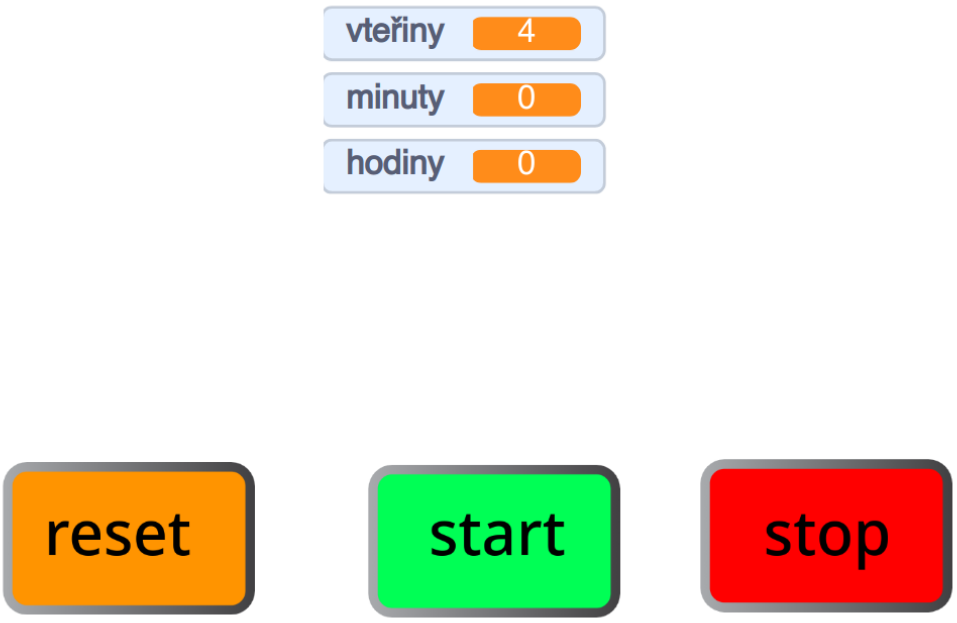
- Práce s proměnnými je pro žáky abstraktní. Je vhodné si dát záležet na vysvětlení práce s proměnnými.

Tabulka 6.7: Soubory pro lekci 12 - Body - proměnné v programu Scratch

Název souboru	Odkaz
Lekce 12 - Body [zadání]	`https://scratch.mit.edu/projects/504656593/`
Lekce 12 - Body [řešení]	`https://scratch.mit.edu/projects/504656690/`

6.3.13 Lekce 13 - Stopky s programem Scratch

V této lekci si žáci vyzkouší naprogramování stopek. Úloha navazuje na předchozí dvě lekce a prohlubuje znalosti v oblasti práce s proměnnými. Žáci si vyzkouší naprogramovat program pro simulování stopování času. Scéna (obrázek 13) obsahuje tlačítka *start, stop a reset*. Výsledný program dokáže se zaokrouhlením na vteřiny počítač čas.



Obrázek 13: Scéna lekce 13

Cíle výuky

- Žák chápe použití příkazů: *po kliknutí na mě a čekej [číslo] sekund*;
- žák chápe principy práce s *proměnnými*, zejména možnosti interakce mezi proměnnými.

Prekoncept

- Žák využívá předcházejících znalostí práce s *proměnnou*;
- žák dokáže přepočítávat vteřiny na minuty a hodiny.

Postup

- Učitel představuje výchozí scénu se třemi postavami (obrázek 13);
- učitel zopakuje koncept proměnné;
- učitel vysvětlí, jakým způsobem se budou nastavovat a zvyšovat hodnoty proměnných;
- učitel představí principy přičítání mezi proměnnými;

- učitel s žáky zopakuje principy počítání hodin;
- učitel předává žákům vstupní soubor (tabulka 6.8).

Poznámky

- Tato lekce byla zařazena pro upevnění znalostí v oblasti práce s proměnnými.

Tabulka 6.8: Soubory pro lekci 12 - Stopky s programem Scratch

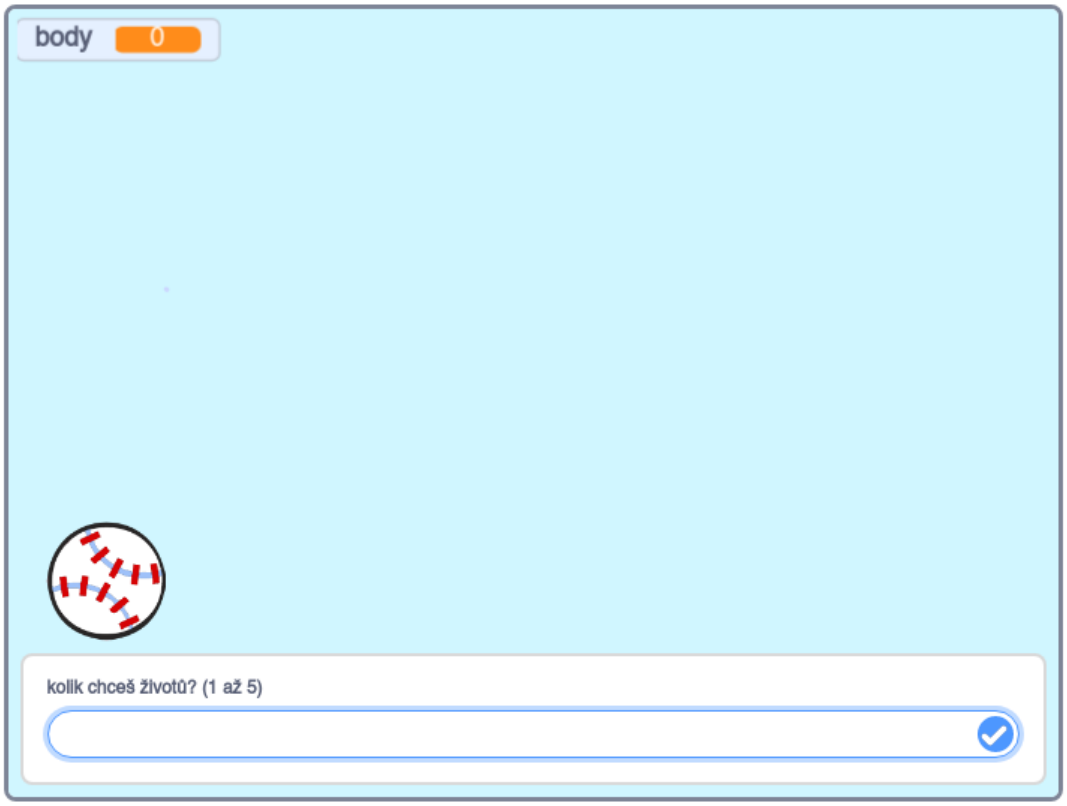
Název souboru	Odkaz
Lekce 13 - Stopky [zadání]	`https://scratch.mit.edu/projects/504610521/`
Lekce 13 - Stopky [řešení]	`https://scratch.mit.edu/projects/504654325/`

6.3.14 Lekce 14 - Klikání na balóny v programu Scratch

Smyslem lekce je zaměřit se na propojování znalostí z předchozích lekcí. Žáci si vyzkouší práci s postavami v kombinaci s počítáním bodů v proměnných. Úkolem žáků je naprogramovat hru v podobně klikání na jednotlivé postavy a počítání bodů resp. životů.

Cíle výuky

- Žák chápe nové algoritmické konstrukty: *načtení uživatelského vstupu, nastav velikost na [číslo] %, otázka, změň velikost o [číslo], přepni pozadí, změň velikost o [číslo], skoč na x: [číslo] y: [číslo], odpověď* a dokáže je využít při programování hry Klikni na balóny;
- žák dokáže pomocí příkazu přepni pozadí na změnit pozadí scény na jiné podle vlastního výběru;
- žák zvládá využití dalšího konstruktu: otázka s uložením vstupní informace do proměnné.



Obrázek 14: Scéna lekce 14

Prekoncept

- Žák využívá znalostí práce s proměnnou z předchozích dvou lekcí;

Postup

- Učitel předá žákům vstupní soubor se scénou (obrázek 14) a několika postavami;
- celkem jsou na scéně tři postavy: fotbalový, basketbalový, baseballový balon, který má po kliknutí na něj zmizet;
- učitel vysvětlí cíl hry, kterým je stihnou kliknout na balony dřív, než dokáží samy zmizet.

Poznámky

- Zpravidla se hra žákům vysvětluje před jejím předvedením.

Tabulka 6.9: Soubory pro lekci 14 - Klikání na balóny v programu Scratch

Název souboru	Odkaz
Lekce 14 - Klikání na balóny [zadání]	`https://scratch.mit.edu/projects/504613789/`
Lekce 14 - Klikání na balóny [řešení]	`https://scratch.mit.edu/projects/504655160/`

6.3.15 Lekce 15 - Tipovací hra s programem Scratch

Lekce s Tipovací hrou je zaměřena na procvičení práce s proměnnými. Nové znalosti se týkají zejména oblasti porovnávání proměnných.

V hotovém programu, který je cílem žáků, hráč zadá 3 různá čísla (obrázek 15) a zkouší jestli se mu podaří trefit do správného trojčíslí. Podle počtu správných čísel postava soba zahlásí výhru.

Obrázek 15: Scéna lekce 15

Cíle výuky

- Žák zkoumá další možnosti práce s proměnnými;
- žák umí využít předcházející znalosti pro programování tipovací hry;
- žák využívá příkazu rovná se pro porovnání dvou proměnných;
- žák rozšiřuje algoritmický konstrukt bublina o načítání hodnoty z proměnné.

Prekoncept

- Žák potřebuje znát práci s proměnnými z předchozích lekcí;
- žák musí chápat porovnání dvou čísel.

Postup

- Učitel představí, jak hra funguje;
- učitel předá žákům vstupní soubor, kde je scéna a tři proměnné;
- učitel vysvětlí způsob porovnání dvou proměnných.

Poznámky

- Žáci si občas pletou jaké proměnné chtějí nastavit uživatelem a jaké losovat automaticky.
- Je vhodné začít s menším počtem tipovaných čísel a poté přidávat další.

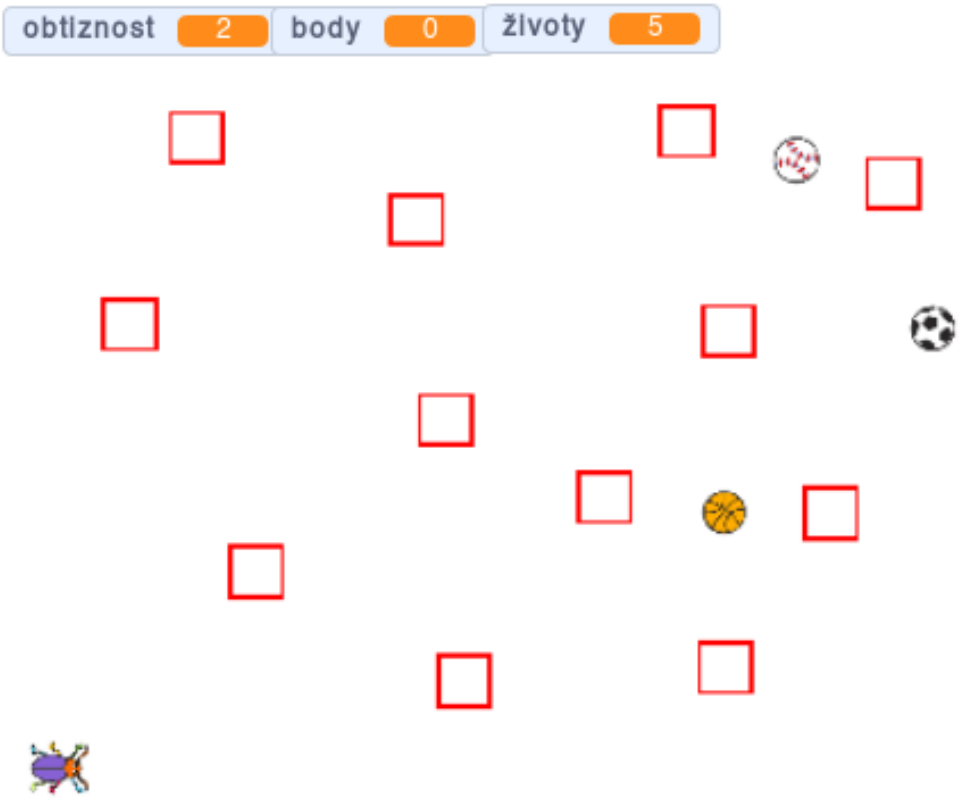
Tabulka 6.10: Soubory pro lekci 15 - Tipovací hra s programem Scratch

Název souboru	Odkaz
Lekce 15 - Tipovací hra [zadání]	`https://scratch.mit.edu/projects/504614075/`
Lekce 15 - Tipovací hra [řešení]	`https://scratch.mit.edu/projects/504655365/`

6.3.16 Lekce 16 - Závěrečná hra v programu Scratch

Závěrečná lekce kombinuje všechny předchozí znalosti v oblasti pohybu postav, jejich pohybu, dotýkání se barev a práce s proměnnými.

Výsledkem snažení žáků by měla být funkční hra, ve které hráč pohybuje postavou brouka a snaží se posbírat co nejvíce balónů. Při pohybu se nesmí dotknout červených políček. Při dotyku s červenou barvou (čtverečkem) ztrácí život. Také se při tomto kontaktu mění kostým brouka. Vstupní soubor pro žáky a výsledné řešení je odkázán tabulce 6.11.



Obrázek 16: Scéna lekce 16

Cíle výuky

- Žák dokáže propojit a použít všechny předcházející poznatky a postupy;
- žák používá nový algoritmický konstrukt: *změň kostým*;
- žák dokáže naprogramovat závěrečnou hru podle zadání vyučujícího.

Prekoncept

- Žák využívá práci s proměnnými využívanou v předcházejících lekcích.

Postup

- Učitel představí výslednou podobu hry pro lepší pochopení zadání;
- učitel předá vstupní soubor obsahující scénu a postavy.

Tabulka 6.11: Soubory pro lekci 16 - Závěrečná hra v programu Scratch

Název souboru	Odkaz
Lekce 16 - Závěrečná hra [zadání]	`https://scratch.mit.edu/projects/504614084/`
Lekce 16 - Závěrečná hra [řešení]	`https://scratch.mit.edu/projects/504652303/`

7 Výsledky

Lekcemi prošli všichni žáci ze skupiny. Je třeba vzít v úvahu, že do kroužku programování chodili žáci z vlastního spontánního zájmu a tudíž nepředstavují reprezentativní vzorek populace stejného věkového rozmezí.

Na závěry hodin je nahlíženo z pohledu jednotlivých aplikací (GalaxyCodr, LigtBot, Blockly games, Scratch) a z pohledů různých aspektů (problémově orientované výuky, gamifikace, konstrukcionistické výuky, tvůrčího myšlení a sociálně stimulujícího prostředí).

7.1 Závěry z výukových lekcí

Závěry z hodin byly formulovány podle jednotlivých aplikací resp. dětských programovacích jazyků.

7.1.1 Aktivity s GalaxyCoder

Tato aplikace je určitě přínosná, bohužel má jen 42 úrovní. Žáci byli schopni se dostat během dvou hodin až do konce. Při využití této hry je třeba počítat s tím, že maximální využití pro všechny žáky lze odhadnout (nejen ty v zájmovém kroužku) na maximálně 3 - 4 vyučovací hodiny. Protože je cílem hry rozvíjet algoritmické myšlení u žáků, pak je tato aplikace vhodná. Žáci nemusí umět základy programování jako takového.

Rozvoj algoritmického myšlení prostřednictvím hry je pro žáky velmi příjemný. Hraní her je pro ně naprosto přirozené. Pokud jsou navíc odměňováni a mohou spolu soutěžit, kam až se ve hře dostali, je to pro ně velmi motivační. Žáci brzy pochopili, že mohou získat více odměn za vyřešení jednotlivých úrovní efektivnějším algoritmem. Ti soutěživější z nich se tak vraceli k přepracování již vyřešených úloh.

Tempo přidávání nových algoritmických konstruktů se zdá jako přiměřené. Jako pro-

blematické se jevilo zařazení algoritmického konstruktu *cyklus* již v úrovni 11. Úroveň 21 byla pro žáky velmi těžká, bylo třeba návodných poznámek učitele. Pro žáky bylo složitější používání systému „lepivých bloků“ kódu v rozhraní aplikace. Často si při přesouvání jednoho bloku omylem rozbili celý program.

Jedinou překážkou by mohl být jazyk aplikace. V tomto případě byla zvolena slovenština. Lze předpokládat, že by žáci neměli problém ani s použitím angličtiny. Naopak by to mohlo být výhodou jako použití metody CLIL.[1]

7.1.2 Aktivity s LightBotem

Tato aplikace je náročnější tím, že postup v ní je daleko rychlejší. Nehodí se pro úplné začátečníky, protože zde použité konstrukty přibývají daleko rychleji. Pokud bude využita pouze 1. lekce, pak žáci vystačí s vlastní intuicí. Pokud by chtěl vyučující pokračovat a využít celou hru, pak bude asi nezbytné žákům vysvětlit, jakým způsobem fungují procedury.

Díky kreslenému průvodci ovládání žáci rychle chápou ovládání programu. Přínosem této aplikace je, že robot musí navíc rozsvěcet žárovku. K doposud požadovanému myšlení přibyl další prvek. Dalším rozšířením je, že žák v této aplikaci musí sám vymyslet celou cestu. Musí si sám zformulovat myšlenky tak, aby splnil celé zadání. Vynikající je rychlá zpětná vazba prostřednictvím animace robota.

Už je vidět, že žáci nad problémem přemýšlí. Dokáží si vzájemně poradit a poznávají, že existuje více správných řešení. Vzhledem k tomu, že se jedná o vybrané žáky, kteří mají o obor zájem, není třeba významně podporovat jejich aktivizaci, jsou sami motivováni. Algoritmické myšlení je rozvíjeno hravou a nenásilnou formou.

[1] Gabriela Baladová et al.: *Výuka metodou CLIL* [online]. 2009 [cit. 02. 04. 2021] Dostupné z: `https://clanky.rvp.cz/clanek/o/z/2965/vyuka-metodou-clil.html/`.

7.1.3 Aktivity s Blockly Games

Další příkaz, který se zde objevil a žáci předtím neznali je *podmínka if a if else*. Novinkou pro ně bylo i omezení v počtu bloků, které mohli použít. Omezený počet bloků vynutil použití *cyklu*. Bohužel *cyklus* způsoboval žákům potíže, protože následně neumožňuje připojení dalšího bloku. Chvíli trvalo, než našli takové řešení, které nevyžadovalo další příkaz.

Na první pohled také žáci neviděli možnost, rozkliknout podmíněný příkaz a vybrat konkrétní podmínku. Těžká pro ně byla úroveň 9. Tam se poprvé objevuje podmínka *if else*. Stačilo však žákům říct, že mohou použít nový příkazový blok. V úrovni 10 už byli zase úspěšní, protože už pochopili využití podmíněného příkazu *if else*. Došlo k dalšímu rozšíření algoritmického myšlení.

7.1.4 Aktivity s programem Scratch

Program Scratch se ukázal jako vhodným nástrojem pro danou skupinu žáků. Drobné problémy se projevily u práce se souřadnicemi a úhly. Žáci ve 4. a 5. třídě ještě nemají dostatečné znalosti z matematiky. Po stručném výkladu a zopakování těchto znalostí se neprojevovaly větší problémy.

Zatímco v předchozích aplikacích žáci řešili hlavolamy, programování v jazyku Scratch je koncipováno jako tvůrčí. Zde je nejvíce umožněn rozvoj algoritmického myšlení.

Z akčního výzkumu vyplynuly následující postřehy. Žáky občas rozptyloval pohyb postavy nakreslené z boku. Jako vhodnější se osvědčilo použití postav znázorněné shora. Drobnou technickou připomínkou je, že při použití dataprojektoru je text v zásobníku příkazů příliš malý. Vyučující musel jednotlivé bloky postupně zvětšovat.

Již v lekci číslo 5 (o 4 lekce dříve) objevili někteří žáci algoritmický konstrukt *cyklus*. Ostatní žáci stále používali pouze *posloupnost*.

V lekci 6 se u některých žáků projevovala nedostatečná představivost. Konkrétně při zpracování úlohy s Pilou bylo pro ně složité představit si správný pohyb postavy.

Žáci, kteří objevili předčasně algoritmický konstrukt *cyklus* mají tendence jej nastavovat na *opakuj stále*. Zablokují tím celý program. Je otázkou pro autory programu Scratch, zda by *nekonečný cyklus* neměl mít limit pro maximální počet opakování.

Za kladné přínosy programování ve Scratch lze považovat možnost snadného rozšíření původního zadání o další postavy. Někteří žáci sami navrhli tuto možnost. Bylo zjištěno, že pro rozvoj algoritmického myšlení je někdy výhodnější žáky nechat experimentovat, než ihned zadávat další úlohy.

Při zavádění nového příkazu *skryj se* někteří žáci zapomněli navázat příkazem *ukaž se*. Měli pak problém postavu najít.

Žáci si postupně zvykali na větší množství postav a k nim přiřazených bloků kódu. Než pochopili tuto filozofii programu Scratch, bylo to pro ně mnohdy nepřehledné.

Častým problém při programování byly nedostatečné znalosti v oblasti matematiky, zejména práce s úhly a kartézskou soustavou souřadnic. Bylo třeba opakovaně zařadit úlohy věnující se specificky těmto oblastem hlavně v lekcích 8 a 10.

V lekci 9 někteří žáci zařadili příkaz *skoč na x: [číslo] y: [číslo]* do *cyklu*. Výsledkem byla nefunkčnost jimi navrhovaného řešení, protože se postava stále vracela na výchozí pozici. Případně se snažili přidat příkaz *zastav všechno* do cyklu. To způsobovalo nefunkčnost celého programu. Dále se projevovaly problémy s identifikací nakopírovaných postav.

Motivace žáků se v průběhu práce s programem Scratch měnila. Bylo vidět a sami žáci to potvrdili, že gamifikované aktivity je motivují daleko více.

Od lekce 11 byly do jednotlivých zadání zařazovány také proměnné. S pochopením práce s nimi neměli žáci potíže.

Celkově lze potvrdit, že rozvoj algoritmického myšlení u všech žáků probíhal plynule. Žáci zvládali stále těžší a těžší úlohy. Přestože byly mezi jednotlivými žáky drobné rozdíly, projevil se obrovský pokrok ve způsobu algoritmického myšlení a představivosti. Byla radost vést žáky, kteří zkoumají nové aplikace a programovací jazyk, aniž by si

uvědomovali, že se učí. Ačkoliv měli žáci představu, že „jen hrají zajímavé hry“, bylo vidět, že dochází k velkým pokrokům v jejich algoritmickém myšlení tak, že to bude mít vliv na jejich budoucí rozvoj.

7.2 Analýza navržených aktivit z hlediska různých aspektů

Na vybrané způsoby rozvoje algoritmického myšlení lze pohlížet různě. V následujících podkapitolách je experimentální část zhodnocena z různých pohledů: problémově orientovaná výuka, orientace na gamifikaci, konstrukcionistické aspekty, tvůrčí myšlení a spolupráce a kolaborace.

7.2.1 Problémově orientovaná výuka

Při této výuce je primárním cílem vyřešit problém, který je zadán jako úkol. Předpokladem výuky je to, že každý jedinec disponuje prostředky k řešení těchto úkolů. Začátkem je identifikace problému, už to je prvním krokem k jeho vyřešení. Tato metoda vyžaduje od žáků odhodlání a ochotu riskovat. Prostřednictvím pokusů a omylů se snažit problém vyřešit. Snažit se, dělat chyby a poučit se z nich. A nakonec být úspěšní. To je příhodný způsob, který vede prostřednictvím vybraných aplikací a dětského programovacího jazyka ke zlepšování algoritmického myšlení.

Výhodou této metody je to, že žáky nutí použít aplikace. Úkolem všech použitých zadání je žáky připravit, vést a podporovat je tak, aby porozuměli způsobu řešení problému. Aby došlo k vyřešení problému a tedy splnění úkolu. Výsledkem bylo vždy napsání kódu, který vede k vyřešení problému a očekávané animaci.

Další výhodou problémově orientované výuky je, že žáci mohou s podporou učitele experimentovat. Mohou se ponořit do programování a zlepšovat si při tom své algoritmické myšlení. Po prvotním seznámení s těmi nejjednoduššími aplikacemi se žáci postupně posouvali dál ke složitějším úkolům. Začali s GalaxyCoder, LightBot a Blockly games.

Dostali se až k programovacímu jazyku Scratch. Problémově orientovaná výuka je klíčovou strategií pro zvyšování algoritmického myšlení. Přidaným bonusem je, že se žáci mohou těšit z vyřešení stále složitějších úkolů.[2]

7.2.2 Orientace na gamifikaci

Z pohledu motivace žáků je ideální využití gamifikace. Pokud jsou žáci správně motivováni, pak se to projevuje tím, že nad zadanými problémy tráví mnohem více času, více se snaží o jejich vyřešení a také mají lepší pocit ze svých výsledků.[3]

Takové prostředí se právě nachází u počítačových her. Pokud ho najdeme i v našich aplikacích, zajistí nám to lepší výsledky při zvyšování algoritmického myšlení. Hlavními prvky je postupné zvyšování obtížnosti a vydávání odměn za plnění úkolů. Žák dostává nepřetržitou zpětnou vazbu o funkčnosti svého algoritmu, sám si volí vlastní cíle v návaznosti na obtížnost úkolu.

Z pohledu výuky využívající gamifikaci jsou vybrané aktivity vyhovující. V aplikaci GalaxyCoder žák sbírá motivační kartičky, hvězdičky a další odměny. I v aplikaci LightBot najdeme motivační prvek a tím je rozsvícení žárovky. Možná že zde není tolik odměn a nenajdeme zde příběh, ale žáci mohou postupovat vlastním tempem a také hned vidí výsledky tvorby svého kódu. V další využité aplikaci BlocklyGames opět nenajdeme příběh. Ve využité části Bludiště jsou spíše minipříběhy. Po aplikaci každé části však vidí žáci ekvivalent kódu v jazyku JavaScript. Mohou cítit, jak se jejich vlastní schopnosti zvyšují právě tím, že vidí výsledek své práce v profesionálním programovacím jazyce. Pozitivní odezva se projevila například v tom, že si žáci sami instalovali použité aplikace i doma na svých zařízeních.

Přestože se aplikace mohou zdát jen jako oddechové aktivity, jedná se o gamifikovanou výuku zlepšující algoritmické myšlení.

[2] Jane Willis. A flexible framework for task-based learning An overview of a task-based framework for language teaching. In: *Oxford* Prabhu 1987 (1996), s. 52–62. Dostupné z: `http://www.intrinsicbooks.co.uk/title_by_title/framework.html`.

[3] Thomas Malone. What Makes Things Fun to Learn? A Study of Intrinsically Motivating Computer Games. In: *Pipeline* 6.2 (1981), s. 50.

Téměř dvě třetiny času se žáci zabývali dětským programovacím jazykem Scratch. I zde najdeme mnoho minipříběhů. Odměnou žákům bylo dosažení řešení. Je však třeba dát pozor na to, aby žáci byli dostatečně motivování hledat jakékoliv řešení vedoucí k očekávanému výsledku. Pokud žáci objevili vlastní „neočekávané“ řešení, pak je třeba je v tom podporovat. Jako pozitivní se ve výuce projevilo to, že žáci nebyli limitováni časem. Každý mohl pracovat vlastním tempem a mohl v klidu svou práci dodělat.

7.2.3 Konstrukcionistické aspekty

Podle autora této teorie Seymoura Paperta[4] je učení práce, tedy budování smysluplného produktu. Nejefektivnější způsob výuky je série praktických činností, které vedou k vytvoření konkrétního produktu. Žák se učí tím, že vytvoří něco nového, pro něj atraktivního.

Právě tvorba programu, který vede k žádoucí animaci je prostředkem pro realizaci konstrukcionistického učení.[5] Využití této metody dává žákům zapomenout, že jsou na půdě škol, že vlastně sami plánují a organizují rozvoj svého algoritmického myšlení. Každý z nich může uplatnit svůj vlastní způsob myšlení a tvorby kódu. Vybrané aplikace poskytují příležitost pro zkoumání a atraktivní učení, při kterém se jako vedlejší produkt zlepšuje algoritmické myšlení. Přestože jsou aplikace zaměřené a připravené pro žáky a učitel je zde jen v roli průvodce, často se i on dostává do role toho, kdo se učí nové věci. V tomto výukovém prostředí mohou žáci růst a budovat svoje algoritmické myšlení. Toto prostředí je založeno na základních znalostech, které si žáci sami přinesli a mohou si ověřovat a realizovat své nápady.

7.2.4 Tvůrčí myšlení

Pro vytvoření kódu je třeba mít tvůrčí myšlení. Jedná se o velmi kreativní práci. Žáci vytvářeli abstraktní pojmy pomocí aplikací a dětského programovacího jazyka.

[4] Papert, *The Children's Machine: Bringing the Computer Revolution to Our Schools, Seymour Papert.*

[5] Rambousek, *Vybrané kapitoly z didaktiky a psychodidaktiky.*

Jednalo se o skvělé cvičení kreativity. Žáci obzvlášť v posledních hodinách kroužku měli absolutní volnost nad tím, jakou cestu pro výsledný efekt zvolit. Nutně museli využít svou kreativitu, museli použít myšlení „out of box“. Právě kreativní tvůrčí myšlení připravuje žáky na víc než jen cestu programátorů. Cílem je, aby v budoucnu mohli čerpat právě z nyní použitých postupů. Aby je využili ve všech aspektech svého života napříč všemi kontexty.

Všechny použité aktivity pomáhaly žákům rozvíjet jejich kreativitu v řešení problémů. Zároveň se jedná o aktivity, které nepředpokládají u učitelů programovací zkušenost. Vyučující mají u většiny aktivit také přístup k online prostředí, kde mohou sledovat pokroky a úspěchy žáků a zároveň představovat různá řešení žáků a vyvolávat diskusi nad nimi.

Rozvoj kritického a tvůrčího myšlení u žáků probíhal při hledání možností, zvažování alternativ a řešení problémů. Žáci se stávali stále lepšími řešiteli problémů.

7.2.5 Spolupráce a kolaborace

Právě vyučující je zde jedním z faktorů, které rozvíjí sociálně stimulující klima ve třídě. Prostřednictvím použitých aplikací a dětského programovacího jazyka vede žáky ke kooperaci, vzájemné pomoci a poradě a k uznání druhého či ocenění jeho názoru. Učitel by zde měl být právě tím pomocníkem, který žáky vede ke zlepšování jejich algoritmického myšlení tak, aby nabyté zkušenosti poté mohli žáci využívat při další výuce a i v běžných situacích, do kterých se během života dostanou. Vyučující by měl podporovat spolupráci a kolaboraci ve třídě tak, že svým chováním bude podporovat situace, které dají příležitost být úspěšný všem žákům.

Žáci často sami nabízeli spolužákům svoji pomoc a radu. Běžně tak docházelo k situacím, kdy aktivnější žáci přecházeli mezi počítači a snažili se poradit spolužákům s řešením.

8 Závěry

V souvislosti s rozvíjením algoritmického myšlení se projevuje i vliv konstruktivistického pojetí výuky. Iniciativa žáků se projevuje v tom, že se sami propracovávají vzdělávacím procesem. Učitelé jsou pouze v roli průvodců a organizátorů. Právě při rozvoji algoritmického myšlení má teorie Seymoura Paperta obrovský význam. Vedlejším produktem metod ke zlepšení algoritmického myšlení je i vznik partnerství mezi žákem a učitelem.

Cílem práce bylo ověřit metody, kterými by bylo možné zlepšit algoritmické myšlení u žáků základní školy. Bylo navrženo 16 výukových lekcí, které pokrývaly různé úlohy při tvorbě algoritmu. V průběhu celého testování bylo využito několik výukových aplikací a programů. Ukázalo se, že je velmi přínosné postupovat od jednodušších aplikací k tvořivým k dětským výukovým programům. Žáci vždy řešili úlohy vlastním tempem a postup od jednodušších úloh ke složitějším byl pro ně přirozený. Již naučené algoritmické konstrukty uměli žáci implementovat v dalších a složitějších úlohách. Jednotlivé metody se ukázaly jako účinné a žáci se naučili, jak postupovat při řešení typických úloh s využitím jednoduchých algoritmických konstruktů. Vedlejším ale nikoliv nepodstatným výsledkem je i zvýšení profesionality učitele daného interakcí s žáky.

Závěrem bylo konstatováno, že došlo ke splnění všech čtyř dílčích cílů.

Díky metodě terminologické a obsahové analýzy byl naplněn cíl $\mathbf{C_1}$ v kapitole 3. Byl analyzován pojem informatické myšlení. Dále byly popsány pojmy algoritmické myšlení a algoritmus. Byly definovány a specifikovány vlastnosti algoritmů. Byla zdůvodněna nutnost algoritmizace učební látky. Dále byly představeny algoritmické struktury.

Prostřednictvím analýzy informačních zdrojů bylo v kapitole 4 dosaženo cíle $\mathbf{C_2}$. Byly charakterizovány různé prostředky používané pro rozvoj algoritmického myšlení. Celkem bylo popsáno pět základních skupin včetně unplugged metod, programovatelných robotů, robotických stavebnic, aplikací pro rozvoj algoritmického myšlení a dětských programovacích jazyků.

S aplikací komparativní analýzy byl splněn i cíl $\mathbf{C_3}$. V kapitole 4.3 byly pomocí komparativní analýzy porovnány podle vybraných hledisek všechny vyjmenované metody. V kapitole 5 byly vybrány nejvhodnější aktivity pro rozvoj algoritmického myšlení pro danou věkovou kategorii. Tyto aktivity byly popsány včetně algoritmických konstruktů v nich použitých.

Akční výzkum pomohl k naplnění cíle $\mathbf{C_4}$. V kapitole 6 byl rozebrán výzkumný projekt, který byl aplikován prostřednictvím akčního výzkumu. Byly popsány jednotlivé výukové lekce, tak jak byly aplikovány. V kapitole 7 byly prezentovány výsledky jednotlivých aktivit. Byly zhodnoceny aktivity s GalaxyCoder, LightBotem, Blockly Games a s programovacím jazykem Scratch. V závěru kapitoly 7 byly použité aplikace zhodnoceny i z dalších aspektů.

Hlavní cíl diplomové práce byl naplněn.

Seznam použitých informačních zdrojů

Baladová, Gabriela et al.: *Výuka metodou CLIL* [online]. 2009 [cit. 02. 04. 2021] Dostupné z: https://clanky.rvp.cz/clanek/o/z/2965/vyuka-metodou-clil.html/(cit. na s. 85).

Barr, By Valerie et al. Bringing Computational Thinking To K12. In: 2.1 (2011), s. 48–54. issn: 21532184. doi: 10.1145/1929887.1929905 (cit. na s. 15).

Berry, Miles. *Computing in the national curriculum: A guide for primary teachers.* 2013. isbn: 9781783391431. Dostupné z: http://www.computingatschool.org.uk/data/uploads/CASPrimaryComputing.pdf (cit. na s. 25).

BOFFIN: *Boffin magnetic* [online]. 2020 [cit. 14. 03. 2021] Dostupné z: https://boffinmagnetic.com/cs(cit. na s. 28).

BotLogic.us Team: *Botlogic.us* [online]. 2019 [cit. 03. 03. 2021] Dostupné z: https://botlogic.us/about(cit. na s. 32).

Chambers, Robert. *Practical action.* 2. vyd. Corwin Press, 2018, s. 190–218. isbn: 1483362132. doi: 10.4324/9781315835815-8 (cit. na s. 55).

CodeCombat Inc.: *Code Combat* [online]. [cit. 25. 06. 2019] Dostupné z: https://codecombat.com(cit. na s. 34).

Cortina, Thomas J. Broadening participation: Reaching a broader population of students through "Unplugged" activities. In: *Communications of the ACM* 58.3 (2015), s. 25–27. issn: 15577317. doi: 10.1145/2723671 (cit. na s. 26).

Department for Education. The national curriculum in England: Framework document. In: *Department for Education* December (2014), s. 264. issn: 17532167. Dostupné z: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/381344/Master_final_national_curriculum_28_Nov.pdf (cit. na s. 25).

Educational Foundation micro:bit: *Micro:bit* [online]. [cit. 14. 03. 2021] Dostupné z: https://microbit.org/(cit. na s. 29).

Furber, Simon. Shut down or restart? The way forward for computing in UK schools. In: *Technology* January (2012), s. 1–122 (cit. na s. 16).

FUTSCHEK, GERALD. Algorithmic Thinking: The Key for Understanding Computer Science. In: *Lecture Notes in Computer Science 4226* (2006), s. 159–168. Dostupné z: https://publik.tuwien.ac.at/files/PubDat_140308.pdf (cit. na s. 17).

FUTSCHEK, GERALD ET AL. Developing Algorithmic Thinking by Inventing and Playing Algorithms. In: *Constructionism* (2010), s. 1–10 (cit. na s. 23).

GALAXYCODR.COM: *Galaxycodr.com* [online]. 2017 [cit. 20. 08. 2019] Dostupné z: https://www.facebook.com/pg/galaxycodr/about/(cit. na s. 32).

GOOGLE INC.: *Blockly-games* [online]. 2020 [cit. 11. 07. 2020] Dostupné z: https://blockly.games(cit. na s. 33).

GOOGLE INC.: *Code with google* [online]. [cit. 01. 03. 2020] Dostupné z: https://edu.google.com/code-with-google/(cit. na s. 34).

HANZALOVA ET AL. Algorithm development and programming at elementary education in the Czech Republic. In: *International journal of education and infomation technologies* 9 (2015), s. 175–179. ISSN: 2074-1316. Dostupné z: http://www.naun.org/main/NAUN/educationinformation/2015/a442008-072.pdf (cit. na s. 10).

HENDL, JAN. *Přehled statistických metod.* 4., rozš. Praha: Portál, 2012. ISBN: 978-80-262-0200-4 (cit. na s. 13).

KEMP, PETER. *Computing in the national curriculum A guide for secondary teachers.* Tech. zpr. 2014, s. 34 (cit. na s. 16).

LANDA, L N. *Algoritmy a učení.* Vyd. 1. Praha: Státní pedagogické nakladatelství, 1973 (cit. na s. 21, 22).

LAWSON, AUBREY. Pedagogy of CS Unplugged : Lessons from Outreach and Education Activities in Computer Science. In: ACM (2017). Dostupné z: https://pdfs.semanticscholar.org/1b18/2ed67a35bc413b34db77df1876c6a367ccb5.pdf (cit. na s. 26).

LEGO® EDUCATION: *LEGO MINDSTORMS* [online]. 2020 [cit. 14. 03. 2021] Dostupné z: https://www.lego.com/cs-cz/themes/mindstorms/about(cit. na s. 30).

LEGO® Education: *WeDo 2.0* [online]. 2020 [cit. 14. 03. 2021] Dostupné z: https://education.lego.com/en-us/products/lego-education-wedo-2-0-core-set/45300#confidence(cit. na s. 30).

Lessner, Daniel. Analysis of Term Meaning Computational Thinking. In: *Journal of Technology and Information* 6.1 (2015), s. 71–88. ISSN: 1803537X. DOI: 10.5507/jtie.2014.006 (cit. na s. 15).

Lessner, Daniel: *Střípky z konference Didinfo 2015: Výuka informatiky v Polsku* [online]. 2015 [cit. 23. 03. 2019] Dostupné z: http://ucime-informatiku.blogspot.com/2015/06/stripky-z-konference-didinfo-2015-2.html(cit. na s. 24).

LightBot Inc: *LightBot* [online]. 2017 [cit. 11. 10. 2020] Dostupné z: https://lightbot.com(cit. na s. 33).

Malone, Thomas. What Makes Things Fun to Learn? A Study of Intrinsically Motivating Computer Games. In: *Pipeline* 6.2 (1981), s. 50 (cit. na s. 89).

Matatalab: *matatalab* [online]. 2020 [cit. 14. 03. 2021] Dostupné z: https://matatalab.com/en(cit. na s. 28).

Merkur Toys s.r.o.: *Robotika /Merkur* [online]. [cit. 07. 03. 2021] Dostupné z: https://eshop.merkurtoys.cz/robotika-c10/(cit. na s. 31).

Mezak, J. et al. Learning scenarios and encouraging algorithmic thinking. In: *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2018 - Proceedings* (2018), s. 760–765. DOI: 10.23919/MIPRO.2018.8400141 (cit. na s. 23).

Microsoft Research: *Kodu Game Lab* [online]. 2020 [cit. 04. 03. 2020] Dostupné z: http://www.kodugamelab.com(cit. na s. 38).

MIT: *Scratch* [online]. 2019 [cit. 12. 10. 2019] Dostupné z: https://scratch.mit.edu(cit. na s. 37).

MIT Media Lab et al.: *About ScratchJr* [online]. 2017 [cit. 03. 03. 2021] Dostupné z: http://www.scratchjr.org/about/info(cit. na s. 37).

MUNI FI: *RoboMise* [online]. [cit. 01. 07. 2019] Dostupné z: https://robomise.cz(cit. na s. 33).

NAKHLEH, LUAY: *What is Algorithmic Thinking?* [online]. 2018 [cit. 09. 08. 2018] Dostupné z: https://www.coursera.org/specializations/computer-fundamentals(cit. na s. 10).

NEZVALOVÁ, DANUŠE. Akční výzkum ve škole. In: *Pedagogika* 53 (2003), s. 300–308. Dostupné z: pages.pedf.cuni.cz/pedagogika/?attachment_id=1944&edmc=1944%0A%0A (cit. na s. 55).

NISHIDA, TOMOHIRO ET AL. New methodology of information education with computer science unplugged. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 5090 LNCS (2008), s. 241–252. ISSN: 03029743. DOI: 10.1007/978-3-540-69924-8_22 (cit. na s. 26).

NÚV: *RVP v oblasti Informatiky a ICT* [online]. 2018 [cit. 16. 03. 2019] Dostupné z: http://www.nuv.cz/t/revize-rvp-ict(cit. na s. 9, 24).

OZOBOT AND EVOLLVE, INC.: *Ozobot* [online]. 2020 [cit. 14. 03. 2021] Dostupné z: https://ozobot.com(cit. na s. 27).

PAPERT, SEYMOUR. An exploration in the space of mathematics educations. In: *International Journal of Computers for Mathematical Learning* (1996). ISSN: 13823892. DOI: 10.1007/BF00191473. Dostupné z: http://www.papert.org/articles/AnExplorationintheSpaceofMathematicsEducations.html (cit. na s. 15).

— The Children's Machine: Bringing the Computer Revolution to Our Schools, Seymour Papert. In: *Bulletin of Science, Technology and Society* 14.4 (1994), s. 227–227. ISSN: 0270-4676. DOI: 10.1177/027046769401400426. Dostupné z: http://journals.sagepub.com/doi/10.1177/027046769401400426 (cit. na s. 36, 90).

PAPERT, SEYMOUR ET AL. Situating constructionism. In: *Constructionism.* 1991 (cit. na s. 15).

RAMBOUSEK, VLADIMÍR. *Vybrané kapitoly z didaktiky a psychodidaktiky.* Praha: Univerzita Karlova v Praze, Pedagogická fakulta, 2014, s. 73. ISBN: 978-80-7290-671-0. Dostupné z: https://uprps.pedf.cuni.cz/UPRPS-476-version1-30_rambousek.pdf (cit. na s. 38, 90).

ROBOBLOQ CO. LTD.: *Q-Dino* [online]. 2020 [cit. 14. 03. 2021] Dostupné z: https://www.robobloq.com/product/Q-dino(cit. na s. 29).

ROBOBLOQ CO. LTD.: *Qobo* [online]. 2020 [cit. 14. 03. 2021] Dostupné z: https://www.robobloq.com/product/Qobo(cit. na s. 28).

RUN MARCO: *Run Marco* [online]. [cit. 20. 06. 2019] Dostupné z: runmarco.allcancode.com(cit. na s. 33).

SALANCI, ĽUBOMÍR. *Didaktika programovania*. České Buďějovice: Jihočeská univerzita v Českých Buďějovicích, Pedagogická fakulta, 2018, s. 32. Dostupné z: https://imysleni.cz/images/vyukove_materialy/JU_Didaktika_PRG.pdf (cit. na s. 34).

— *EasyLogo – discovering basic programming concepts in a constructive manner*. 2010, s. 10. Dostupné z: http://www.salanci.sk/EasyLogo/Paper.pdf (cit. na s. 36).

SGP: *Multimediální tvůrčí systém SGP Baltík 3* [online]. [cit. 04. 03. 2021] Dostupné z: https://www.sgpsys.com/cz/DescriptionB3.asp(cit. na s. 35).

SKALKA, JÁN ET AL. *Algoritmizácia a úvod do programovania*. Nitra: UNIVERZITA KONŠTANTÍNA FILOZOFA V NITRE, 2007, s. 158. ISBN: 9788080942175 (cit. na s. 18–22).

SKOLSKE.SK, PORTÁL: *Princípy programovania sa môžu učiť už druháci a tretiaci na ZŠ* [online]. 2017 [cit. 20. 05. 2020] Dostupné z: https://www.skolske.sk/clanok/32395/principy-programovania-deti(cit. na s. 32).

TTS GROUP LTD. *Bee-Bot Rechargeable, chield friendly, programmable fllor robot*. Nottinghamshire: TTS Group Ltd., 2015, s. 8. Dostupné z: https://resources.terrapinlogo.com/robots/bee-bot/user_guide_and_technical_specs.pdf (cit. na s. 27).

UNPLUGGED, CS: *CS Unplugged* [online]. 2020 [cit. 05. 11. 2020] Dostupné z: https://csunplugged.org/en/how-do-i-teach-cs-unplugged/(cit. na s. 25).

VANÍČEK, JIŘÍ ET AL. *Programování ve Scratch pro 2. stupeň základní školy*. České Buďějovice: Jihočeská univerzita v Českých Buďějovicích, Pedagogická fakulta, 2020, s. 17. ISBN: 978-80-7394-783-5. Dostupné z: https://imysleni.cz/

ucebnice/programovani-ve-scratchi-pro-2-stupen-zakladni-skoly (cit. na s. 63).

VEX Robotics, Inc. *VEX IQ - Uživatelská příručka řídicího systému.* Dostupné z: https://www.veskole.cz/downloads/VEX/VEX_IQ_Uivatelska_piruka_idiciho_sytemu.pdf (cit. na s. 31).

Virius, Miroslav. *Základy algoritmizace.* Dot. 1. vy. Praha: České vysoké učení technické, 1997. isbn: 80-01-01346-4. Dostupné z: http://www.rudisweb.wz.cz/dokumenty/algoritmizace.pdf (cit. na s. 18).

Štátny pedagogicky ústav. *Informatika - primárne vzdelávanie.* Tech. zpr. Štátny pedagogický ústav, 2014, s. 1–10. Dostupné z: http://www.statpedu.sk/sk/svp/inovovany-statny-vzdelavaci-program/inovovany-svp-1.stupen-zs/matematika-praca-informaciami/ (cit. na s. 24).

— *Štátny Vzdelávací Program.* Tech. zpr. 2008. Dostupné z: http://www.statpedu.sk/files/articles/dokumenty/statny-vzdelavaci-program/informaticka_vychova_isced1.pdf (cit. na s. 10).

Willis, Jane. A flexible framework for task-based learning An overview of a task-based framework for language teaching. In: *Oxford* Prabhu 1987 (1996), s. 52–62. Dostupné z: http://www.intrinsicbooks.co.uk/title_by_title/framework.html (cit. na s. 89).

Wing, Jeannette M. Computational Thinking. In: *Commun. ACM* 49.3 (2006), s. 33–35. issn: 0001-0782. doi: 10.1145/1118178.1118215. Dostupné z: http://doi.acm.org/10.1145/1118178.1118215 (cit. na s. 15).

Znám Štefan, Bukovský Lev, Milan Hejný Jozef Hvorecký Beloslav Riečan. *Pohľad do dějín matematiky.* Bratislava: Alfa, 1986 (cit. na s. 17).

Seznam obrázků

Seznam tabulek