Využití algoritmických nástrojů ve výuce programování

Bc. Kristýna Šimková

Diplomová práce
2015

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2014/2015

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Kristýna Šimková**
Osobní číslo:	**A13466**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Učitelství informatiky pro střední školy**
Forma studia:	**prezenční**
Téma práce:	**Využití algoritmických nástrojů ve výuce programování**
Téma anglicky:	**The Use of Algorithmic Tools for Teaching Programming**

Zásady pro vypracování:

1. **Zhodnoťte současnou kvalitu výuky programování na středních školách.**
2. **Analyzujte používané algoritmické nástroje pro výuku programování na středních školách.**
3. **Zvolte kritéria, která ovlivňují výuku programování, a pomocí nich vyberte nejvhodnější algoritmický nástroj.**
4. **Vytvořte odpovídající metodické listy, jejichž zadání bude mít stoupající obtížnost.**
5. **Realizujte vytvořené metodické listy ve výuce.**
6. **Vyhodnoťte výsledky realizace.**

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **VLADIMÍRA SEHNALOVÁ, Anna Závadská. Metodika výuky informatiky na 2. stupni základních škol a středních školách z pohledu pedagogické praxe – náměty pro začínajícího učitele. 1. vyd. Ostrava: Ostravská univerzita v Ostravě, 2010. ISBN 978-807-3688-912.**
2. **VIRIUS, Miroslav. Základy algoritmizace. Vyd. 2., přeprac. Praha: Česká technika – nakladatelství ČVUT, 2008. ISBN 978-800-1040-034.**
3. **ANDREJ BLAHO, Ivan Kalaš a [překlad Jiří VANÍČEK]. Imagine Logo: učebnice programování pro děti. Vyd. 1. Brno: Computer Press, 2006. ISBN 80-251-1015-X.**
4. **RESNICK, M., Scratch.mit.edu [online] [cit. 2015-01-12]. Getting Started Guide. Dostupné z WWW: http://download.scratch.mit.edu/ScratchGettingStartedv14.pdf**
5. **BOTEK, Zdeněk. Základy informačních technologií. Vyd. 1. Zlín: Univerzita Tomáše Bati ve Zlíně, 2013. ISBN 978-807-4543-135.**
6. **PECINOVSKÝ, Rudolf. Hrajeme si s Baltíkem – učebnice programování pro děti i jejich rodiče. SGP, 2000.**
7. **PECINOVSKÝ, Rudolf. Učebnice programování – základy algoritmizace: Učebnice s příklady v Turbo Pascalu a Borland C. 1. vyd. Praha: Grada Publishing, 1997, 177 s. ISBN 80-716-9577-7.**

Vedoucí diplomové práce: **doc. RNDr. Zdeněk Botek, CSc.**
Ústav krizového řízení

Datum zadání diplomové práce: **6. února 2015**

Termín odevzdání diplomové práce: **15. května 2015**

Ve Zlíně dne 6. února 2015

L.S.

doc. Mgr. Milan Adámek, Ph.D.
děkan

doc. Mgr. Roman Jašek, Ph.D.
ředitel ústavu

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové/bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně 11.5.2015

.....
podpis diplomanta

ABSTRAKT

Teoretická část diplomové práce hodnotí současnou kvalitu výuky programování na středních školách. Dále zpracovává porovnání algoritmických nástrojů a následně dle kritérií vybírá z nich nejvhodnějšího adepta pro základní výuku algoritmizace studentů středních škol. Praktická část obsahuje mnou vypracované metodické listy, které mohou posloužit jako opora řadě učitelům, kteří budou mít zájem tuto oblast realizovat ve své výuce. Metodické listy mají stoupající obtížnost a logicky na sebe navazují. Závěrečná část diplomové práce se věnuje realizaci metodických listů ve výuce a s tím souvisejícím zhodnocením.

Klíčová slova:

Algoritmické myšlení, metodické listy, programování, Scratch, střední škola

ABSTRACT

The theoretical part of the thesis evaluates the current quality of teaching programming in high schools. Further processing algorithmic comparison tools, and subsequently by criteria selects the most appropriate of these aspirant for basic training algorithmisation high school students. The practical part includes methodology sheets developer by me that can serve as support for a number of teachers who will be interested in this area to implement in their teaching. Methodological sheets have difficulty rising and logically connected to each other. The final part of the thesis deals with the realization of methodological papers in the classroom and the related evaluation.

Keywords:

Algorithmic thinking, methodological sheets, programming, Scratch, high school

Chtěla bych poděkovat všem, kteří jakkoliv přispěli ke zpracování mé diplomové práce. Především vedoucímu práce panu doc. RNDr. Zdeňku Botkovi, CSc. za odborné náměty, připomínky a konzultace, které mi během zpracování diplomové práce poskytl.

Prohlašuji, že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD **8**

I. TEORETICKÁ ČÁST **9**

1 SEZNÁMENÍ S PROBLEMATIKOU **10**

1.1 Algoritmizace a programování 10

1.2 Kvalita výuky programování na středních školách 10

2 POROVNÁNÍ ALGORITMICKÝCH NÁSTROJŮ **14**

2.1 Imagine Logo 14

2.2 Karel 17

2.3 Lego Mindstorms 18

2.4 Baltík 19

2.5 Scrach 21

2.6 Kritéria hodnocení a výběr 22

II. PRAKTICKÁ ČÁST **26**

3 METODICKÉ LISTY PRO VÝUKU PROGRAMOVÁNÍ **27**

3.1 Metodický list 1 27

3.2 Metodický list 2 31

3.3 Metodický list 3 34

3.4 Metodický list 4 39

3.5 Metodický list 5 43

3.6 Metodický list 6 48

3.7 Metodický list 7 54

3.8 Metodický list 8 57

4 REALIZACE A VYHODNOCENÍ **62**

4.1 Realizace metodického listu č. 1 62

4.2 Realizace metodického listu č. 2 63

4.3 Vyhodnocení realizace metodických listů 63

ZÁVĚR 64

SEZNAM POUŽITÉ LITERATURY.....66

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....68

SEZNAM OBRÁZKŮ 69

SEZNAM GRAFŮ 71

SEZNAM PŘÍLOH.....72

ÚVOD

Lidská historie popisuje svá období podle dominantního využití nástrojů a řadou tisíců let dospěla od doby kamenné do současné doby, která se dá s lehkou nadsázkou označovat jako doba počítačová. Bezpochyby totiž dnešnímu světu vládnou různé moderní informační a komunikační technologie a zdá se, že tento trend je nezastavitelný a neustále zvyšujícím se tempem se hrne kupředu. Důležitý aspekt současného vzdělávání by tedy měl být kladen především na podporu dostatečného seznámení studentů s moderními technologie a schopností efektivní práce s počítačem.

Střední škola je místo, které ze značné části ovlivňuje výběr budoucího povolání a tudíž je zcela na místě se zamyslet, zda právě výuka algoritmizace by neměla mít v školním vzdělávacím programu více místa. Jelikož existuje stále velké množství základních škol, kde se objevuje jakákoliv absence výuky základů algoritmického myšlení a tudíž mnohdy až na střední škole se s tímto oborem studenti seznamují. Negativum tohoto postupu je zcela jistě v tom, že studenti postrádají základy algoritmického myšlení a přeskočí ihned na následující krok, který s těmito znalosti počítá a to je zápis algoritmů ve vybraném programovacím jazyce. Strohý zápis v programovacím jazyce bez hlubšího vysvětlení může být tedy pro řadu žáků značně demotivující. Výjimku budou pravděpodobně tvořit pouze ti studenti, kteří sami od sebe prostoupili do tajů programování a vytvořili z nich svůj vlastní koníček.

Ovládaní problematiky algoritmů je nejen důležité pro budoucí programátory a IT specialisty, ale i pro ostatní studenty, jimž její zvládnutí zajistí rozvoj logického, abstraktního myšlení a donutí je zpracovávat problémy pomocí dílčích kroků, které si při obdržení problému předem nastíní a tím efektivněji dosáhnou tíženého výsledku.

Jedním z cílů mé diplomové práce je zhodnotit současnou kvalitu výuky programování na středních školách. Dalším dílčím cílem je pak zpracování porovnání algoritmických nástrojů - dle kritérií následuje výběr nejvhodnějšího z nich a následné ověření jeho atraktivity ve výuce pomocí vypracovaných metodických listů, které budou obsahovat zadání úloh se stoupající obtížnost. Zadání bude tvořeno tak, aby jednodušší úlohy bylo možno naprogramovat za jednu vyučovací hodinu. Závěrečné metodické listy budou obsahovat i složitější úlohy, které se budu snažit rozložit do jednotlivých částí, jejichž funkci bude možné samostatně otestovat. Mezi další dílčí cíle patří ověření, zdali použití metodických listů vypracovaných v rámci této práce, povede ve výuce k lepšímu pochopení dané problematiky žáky a tím ke zlepšení celé výuky.

I. TEORETICKÁ ČÁST

1 SEZNÁMENÍ S PROBLEMATIKOU

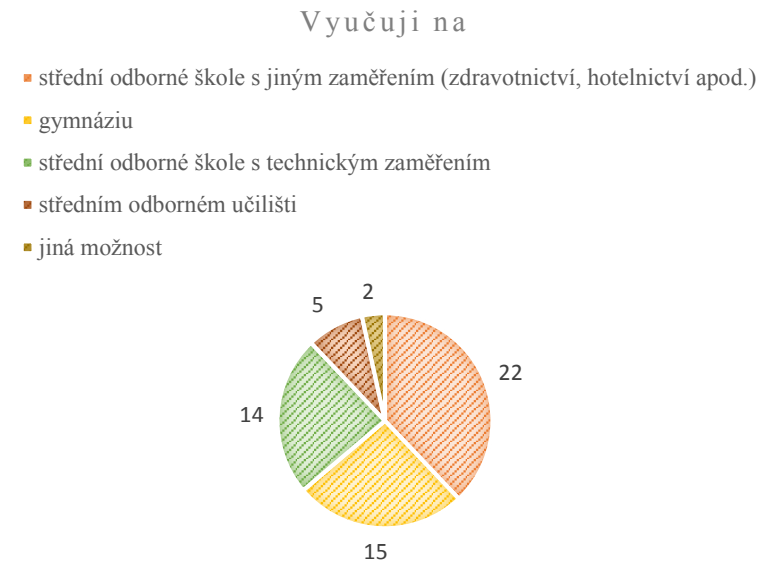
První podkapitola představuje základní pojmy v obecné rovině, přičemž další podkapitola předkládá výsledky vypracovaného dotazníku a tím dává čtenáři povědomí o míře kvality výuky algoritmizace a programování na středních školách.

1.1 Algoritmizace a programování

Algoritmizaci lze označit jako zcela tvůrčí činnost, jejímž cílem není pouze najít postup, který povede k řešení daného problému, ale také postup, který bude relativně krátký a především rychlý. Jak uvádí na portálu Česká škola docent Jiří Vaníček, vedoucí katedry informatiky Pedagogické fakulty Jihočeské univerzity v Českých Budějovicích, cílem výuky algoritmizace a programování na střední škole bez rozšířeného technického zaměření není zcela jistě vychovat úspěšné programátory (stejně jako cílem výuky matematiky není vychovat budoucí matematiky), ale spíše je cílem trénovat řadu kompetencí z oblasti algoritmizace jako je oddálení vykonání příkazu, následné ladění programu, dekompozice problému apod. Student tím získá informatický pohled na počítač, rozvíjí svou tvořivost a také si vyzkouší projektový způsob práce [1].

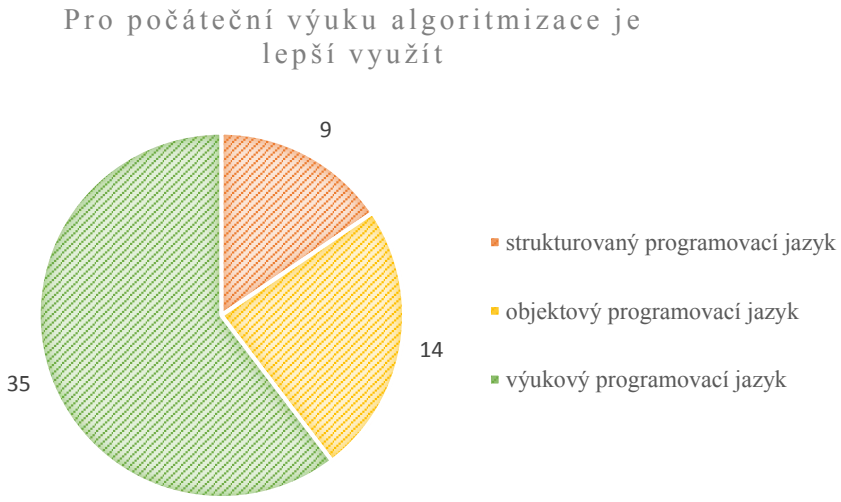
1.2 Kvalita výuky programování na středních školách

Pro objektivní zhodnocení situace jsem vyhotovila dotazník pomocí služby Google formuláře, následně jsem využila sociálních sítí a tento dotazník vyvěsila na dostupných stránkách pedagogických fakult, kde se vyskytuje nejvíce lidí ze školní praxe. Výhodou tohoto způsobu rozšíření dotazníku je ten fakt, že uživatelé sociálních sítí jsou většinou silně nakloněni ke sdílení a také k zpracování informací. Dotazník začíná otázkou, kde pedagog vyučuje a dále pokračuje navazujícími, více specifickými otázkami – kompletní dotazník je k nahlédnutí v příloze této diplomové práce. Dotazník vyplnilo 58 respondentů, což na tak úzce specializovanou oblast lze jistě požadovat za uspokojivý výsledek. Výsledky tohoto šetření ukazují, že 38 % respondentů vyučuje na střední odborné škole s netechnickým zaměřením, kam můžeme zařadit školy zaměřené na zdravotnictví, hotelnictví apod. Dále 26 % vyučuje na gymnáziu a jen o 2 % méně respondentů vyučuje na střední odborné školy s technickým zaměřením. Zbývající procenta zabrali učitelé, kteří vyučují na středních odborných učilištích nebo na jiným blíže nespecifikovatelných typech škol.



Graf 1 – Typ střední školy

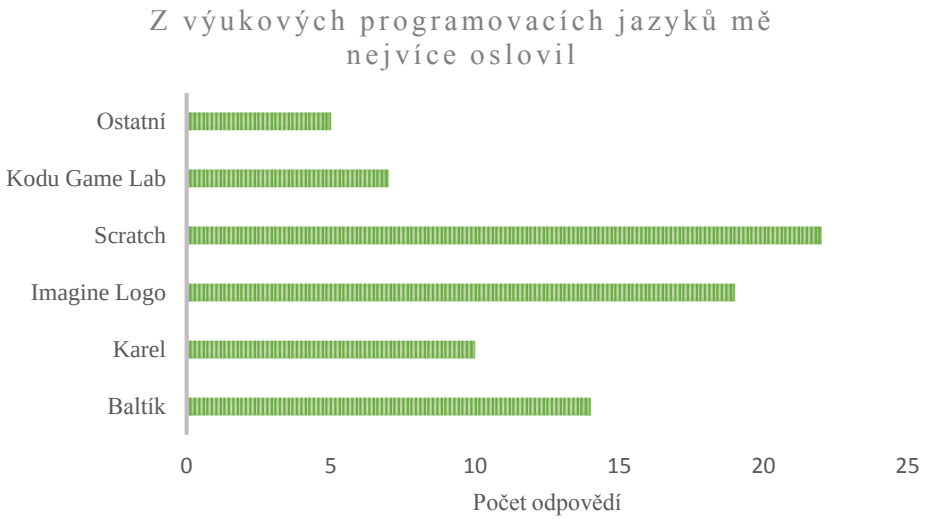
Dále z výzkumného šetření vyplynulo, že nejvíce učitelů pro počáteční výuku algoritmizace preferuje výukový programovací jazyk a až v dalekém závěsu za tímto názorem se objevuje skupina učitelů domnívajících se, že je v tomto případě vhodnější využít objektový či strukturovaný programovací jazyk.



Graf 2 - Počáteční výuka algoritmizace

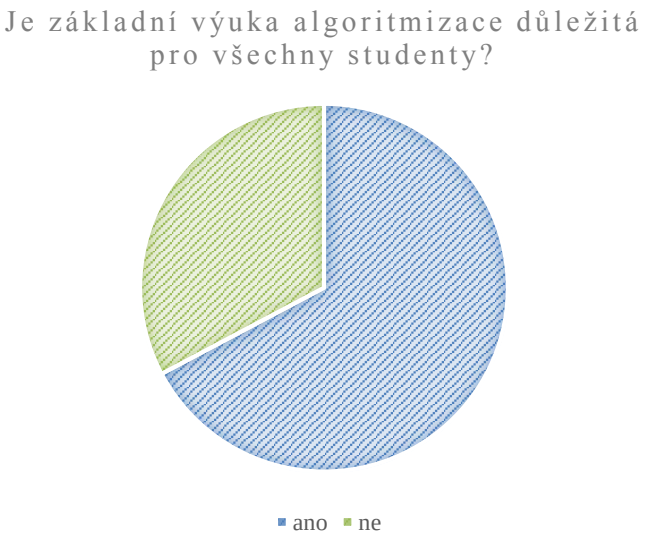
Po povinných otázkách následovaly dvě spolu spjaté, jejíž odpovědi nebyly přímo vyžadovány, a to *Který programovací jazyk ve výuce používáte převážně?* a *A z jakého důvodu?* Odpovědi byly velmi rozmanité, od velmi populárních programovacích jazyků jako je Java, PHP, C, C# až po ty méně známé jako je například Ruby. Za zmínku zcela jistě stojí, že programovací jazyk Ruby letos oslavil dvacet let od svého vzniku a stále se vyvíjí a pravděpodobně i z toho důvodu

se objevil v odpovědích dokonce pětkrát. Je to plně objektově orientovaný jazyk, který je díky jednoduché syntaxi poměrně snadný k naučení, přesto však neztrácí na svém výkonu a tak může konkurovat známějším jazykům (Python, Perl a jiné) [2]. Pokud by se má diplomová práce nezaobírala výukovými programovacími jazyky, zcela jistě při tvorbě metodických listů by má volba padla na tento perspektivní programovací jazyk. Mezi nejčastější důvod volby vybraného programovacího jazyku patřil ten fakt, že učitelé s ním mají již předchozí znalosti a právě proto jej používají i ve výuce. Další otázky byly již více zaměřené na problematiku ohledně výukových programovacích jazyků.



Graf 3 - Výukové programovací jazyky

Dle dotazníků mezi výhody výukových programovacích jazyků patří jejich vizuální odezva a pochopení základních principů algoritmů. Další otázka se zaměřovala na specifickou výuku algoritmizace pomocí programovatelné robotické stavebnice Lego Mindstorms, kde více než polovina učitelů uvedla, že ji nevyužívá zejména z důvodu velké počáteční investice. Dále respondenti uváděli, že většinu začátečníků od programování odradí nejčastěji složitost pochopení struktury kódu, nutnost dodržování přísné syntaxe a v neposlední řadě složitost vývojového prostředí. Závěrečná otázka se ptala, zdali je základní výuka algoritmizace důležitá pro všechny studenty?



Graf 4 Algoritmizace pro všechny studenty

Mezi nejčastější uváděné důvody proč učit algoritmizaci všechny studenty patří tyto odpovědi: algoritmizace rozvíjí logické myšlení, naučí studenty lépe chápat, jak funguje počítač a umožní lepší pohled na řešení problémů. Nejvíce mne zaujala odpověď, že algoritmizace a s tím související programovaní je vlastně další způsob myšlení a pokud má smysl vyučovat na všech typech škol matematiku na základní úrovni, tak má stejný smysl takto vyučovat algoritmizaci. Z dotazníkového průzkumu tedy vyplývá, že respondenti jsou pro to, aby se základy algoritmizace naučil každý student například bez ohledu na jeho budoucí pracovní cíle. Dále skoro dvě třetiny dotázaných si myslí, že pro základní výuku algoritmizace je nejvhodnější volit některý z výukových programovacích jazyků, což lze brát jako motivaci pro vytvoření odpovídajících metodických listů.

2 POROVNÁNÍ ALGORITMICKÝCH NÁSTROJŮ

Následující podkapitoly popisují nástroje, které spadají do kategorie pro výuku algoritmizace. Nástroje jsou nejprve obecně popsány a poté v poslední podkapitole zhodnoceny dle kritérií, které usnadní následný výběr toho nejvhodnější z nich, který bude vybrán k následné tvorbě metodických listů.

2.1 Imagine Logo

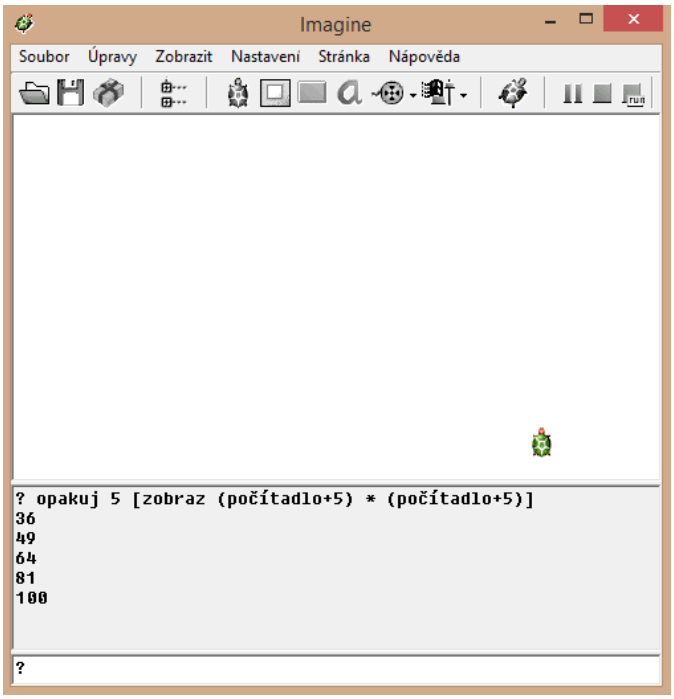
Prostředí Imagine Logo je moderní didaktický nástroj pro výuku programování na ZŠ i SŠ, které čerpá ze svého pověstného předchůdce, jenž řada škol hojně v počátku 90. let používala, a to je Comenius Logo. Nástroj Imagine Logo se opírá o ověřenou generaci svých předchůdců, postavených na programovacím jazyku Logo a podporuje konstruktivistickou teorii učení J. Piageta, která vychází z toho, že když se člověk učí, nevstřebává a neosvojuje si nové porozumění pouze pasivně. Naopak, nové informace se aktivně integrují do dosavadní známých struktur a jsou pochopeny prostřednictvím těchto schémat, které člověk má, ale současně je může také přetvářet. Proto je vše, co se člověk učí zasazeno do kontextu toho, co už předem ví. Tím pádem si každý z nás si prostřednictvím interakcí vytváří vlastní způsoby, struktury porozumění světu [3]. Technická stránka prostředí je postavena na objektově orientované struktuře – podporuje tedy hierarchii objektů, děditelnost vlastností a také objektové proměnné [4]. Metodika výuky algoritmizace spojená s představivostí při řešení problémů pomocí kreslení v tzv. želví geometrii je navíc ve světě dostatečně otestována [1]. Uživatel při programování ovládá tedy grafický kreslící objekt – želvu, která je vytvořena přímo pro české prostředí a reaguje na základní příkazy vlevo, vpravo, dopředu, smaž. Umožnuje také využití cyklů a podmínek, a to prostřednictvím příkazů opakuj a když. Pomocí příkazů psaných do příkazového řádku lze želvu tedy ovládat a ihned vidět její kroky, prostřednictvím nakreslených čar na obrazovce. Prostředí umožňuje i psaní jednoduchých procedur do tzv. paměti želvy. Studentům, tak umožní přirozeným způsobem a poměrně rychle se seznámit i s tvorbu jednoduchých her, což zcela jistě tomuto nástroji přidá na atraktivitě. Obzvlášť ta skutečnost, že prostředí Imagine Logo umožňuje vytvářet programy, které pracují síťově – běží současně na více počítačích v síti a komunikují spolu, což umožňují studentům vytvářet vlastní internetové chaty či lákavé síťové hry. Výhodou programovacího jazyku Logo je také podpora důležité programátorské techniky a to rekurze, což není považováno zcela za samozřejmost. Příkladem absence rekurze může být například první široce používaný programovací jazyk Fortran [5]. Rekurze je metoda zápisu algoritmu, která volá sama sebe, přičemž mezi její výhody patří především kratší zápis kódu a

význam má hlavně rekurze s návratem – metoda pokusů, v případě omylu se vrátíme vždy o krok zpět [6].

Ukázky úloh:

Matematické operace

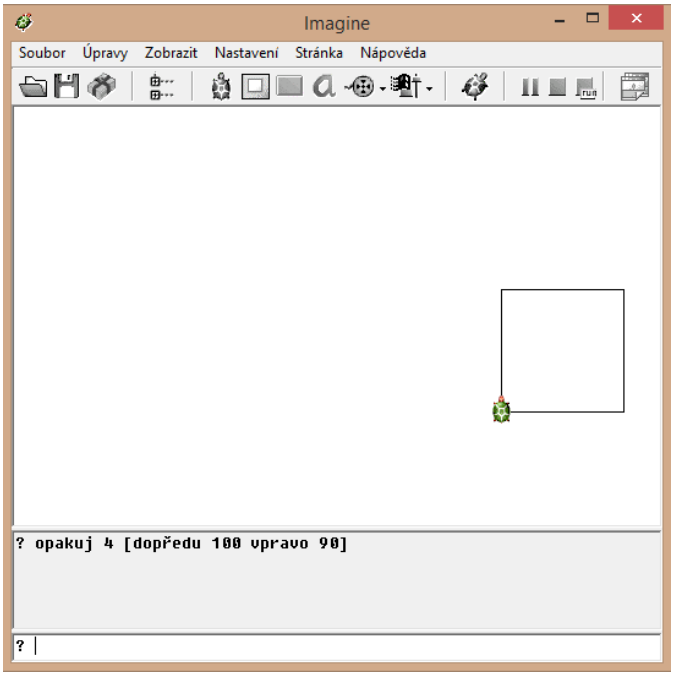
Napište příkaz, kterým vypíše Imagine Logo řadu následujících čísel: 36 49 64 81 100



Obrázek 1 Imagine Logo - Řada čísel

Jednoduché grafické úlohy

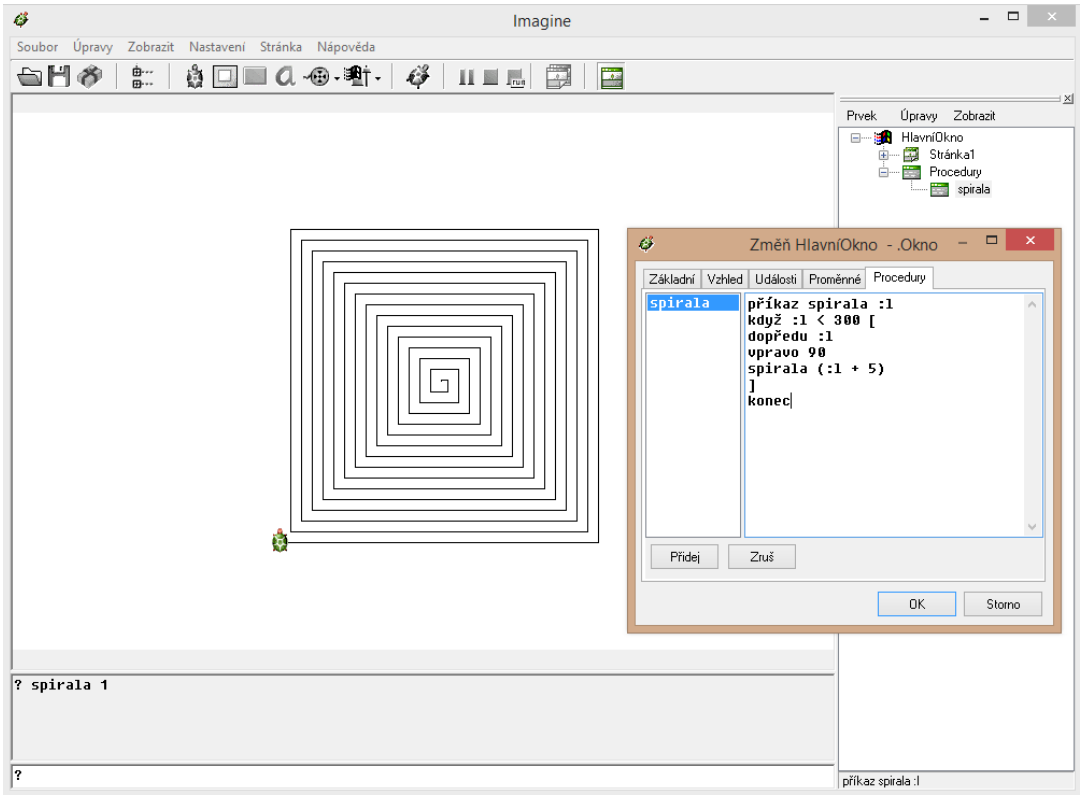
Použijte příkaz cyklu pro nakreslení čtverce.



Obrázek 2 Imagine Logo - Příkaz cyklu

Složité grafické úlohy

Sestavte rekurzivní proceduru pro nakreslení spirály.

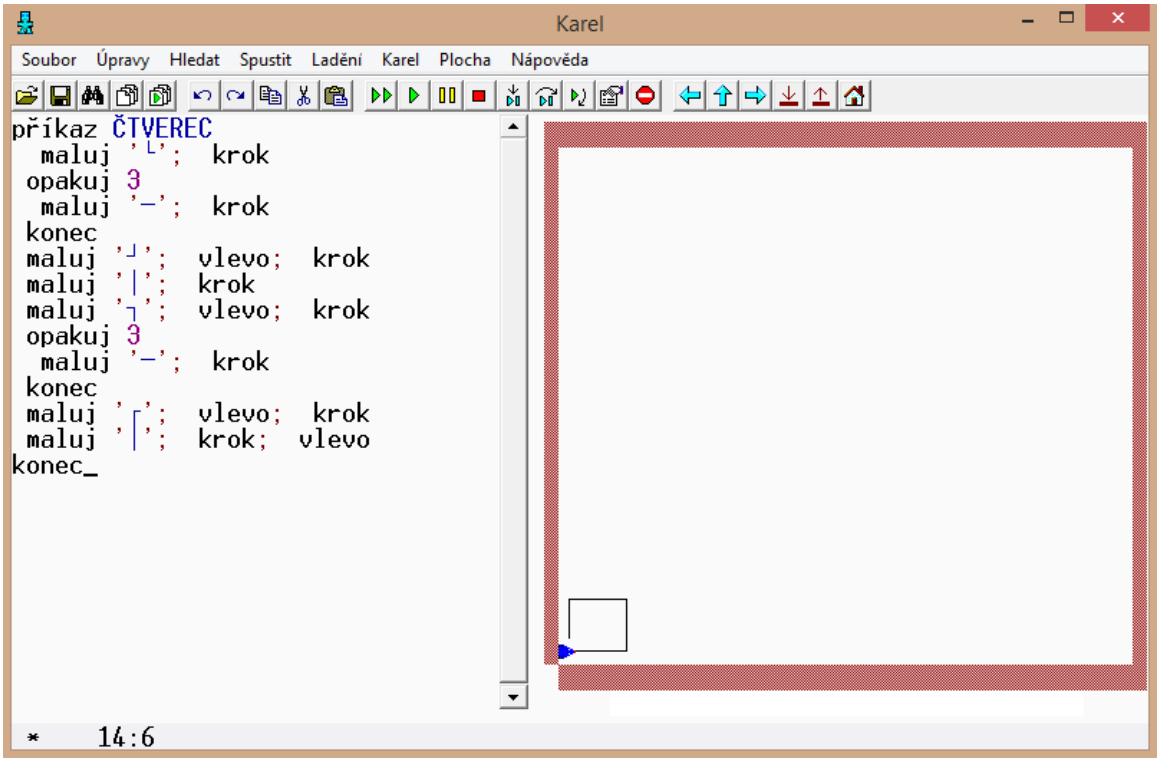


Obrázek 3 Imagine Logo - Rekurzivnost procedur

2.2 Karel

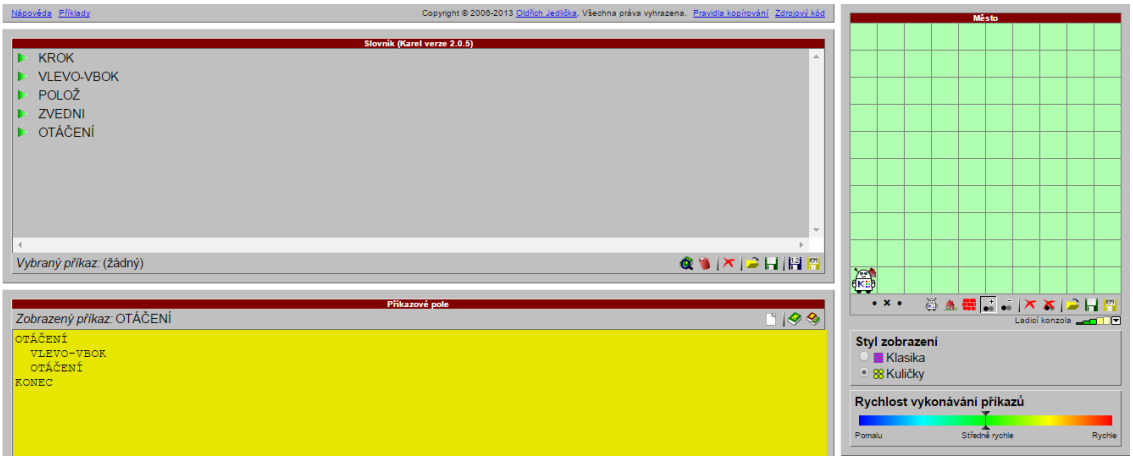
Richard E. Pattis, profesor ze Stanfordské univerzity ve Spojených státech, přišel počátkem 70. let 20. století na myšlenku vytvořit programovací jazyk určený pro výuku programování absolutních začátečníků. Následně tuto myšlenku realizoval, publikoval knihu Karel the Robot: A gentle introduction to the art of programming a s tím je spojen vznik programovacího jazyku Karel, čímž autor vzdal tak hold Karlu Čapkovi, tvůrci slova robot, jenž je hlavní postavou tohoto programovacího jazyku [7]. Původní Karel se neustále vyvíjí a tím nejzásadnějším zvratem je zcela jistě rok 2005, kdy se Karel objevil v nové implementaci v programovacím jazyce Java spolu s open source vývojovou platformou Eclipse. Dle publikace Stranforské univerzity studenti, jenž začínají v programovacím jazyku Karel, poté plynule navazují v průběhu několika týdnů na vyšší programovací jazyk Java, což v dnešní době nesporně patří mezi značnou výhodu [8].

Robot Karel existuje v současné době buď v atraktivní a hlavně zcela intuitivní on-line verzi http://karel.oldium.net/ nebo v off-line verzi 4.2, která umožňuje navíc při programování používat celočíselné proměnné, vícerozměrná pole, procedury a funkce s parametry a v neposlední řadě základní aritmetické a logické operace. Tvůrce přiloženou dokumentací udává, že verze je vhodná hlavně k procvičení používání procedur a rekurzivních algoritmů [9].



Obrázek 4 Karel – Příkaz čtverec

On-line verze z hlediska algoritmických vlastností umožňuje jak začátečnické skládání příkazů, rekurzi až po složitější vnořené podmínky. Názorné vytvoření jednoduché rekurze v on-line verzi.

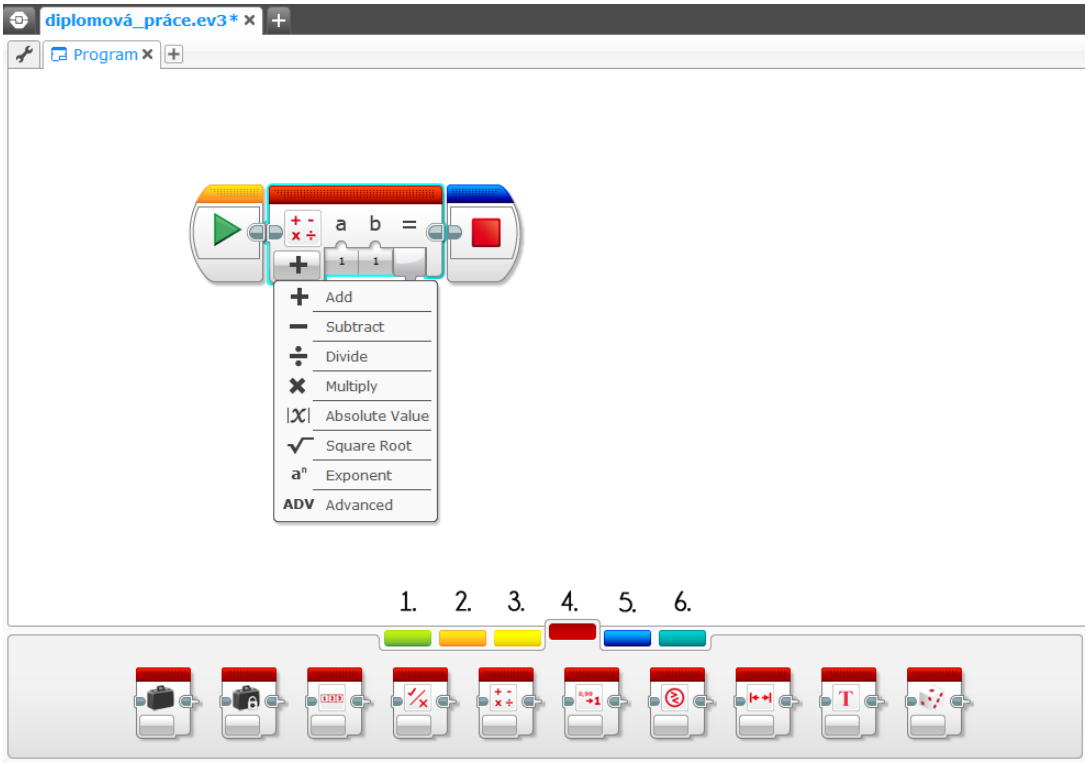


Obrázek 5 Karel - Jednoduchá rekurze

2.3 Lego Mindstorms

Stavebnice Mindstorms vyráběná firmou Lego, se řadí k programovatelným robotickým stavebním sadám, které zábavnou formou umožňují studentům ovládat své vlastní roboty pomocí naprosto intuitivního softwarového programu, včetně nejnovější verze EV3, která dokonce podporuje ovládání prostřednictvím smartphonu. Software obsahuje široké množství prostředků pro snadno pochopitelnou základní výuku algoritmizace, kde sestavování algoritmů je tvořeno spojením jednotlivých grafických částí, které znázorňují příkazy, podmínky a následně i cykly. Tudíž odpadá nutnost znalosti jakékoliv syntaxe, což podporuje především lepší soustředěnost studentů na princip řešení daného problému a ne na formu zápisu. K dispozici jsou následující bloky:

1. Akční bloky - ovládají otočky motoru, zvuk, světlo
2. Funkční bloky – obsahují jednotlivé funkce programu – cyklus, přerušení cyklu
3. Senzorové bloky - umožňují programům číst informace přicházející ze senzorů
4. Operační bloky – slouží k psaní, čtení proměnných a porovnávání jejich hodnot
5. Pokročilé operace – spravují soubory, posílají zprávy a slouží k připojení přes bluetooth
6. Vlastní operace – vlastnoručně vytvořené bloky



Obrázek 6 Lego Mindstorms - Bloky akcí

Programování robotů není omezeno přímo na software EV3 od firmy Lego, ale lze využít také léty osvědčenou klasiku v oblasti grafických programovacích jazyků a to LabVIEW [10]. Další software Robotc, založený na programovacím jazyku C, používaný na programování fyzických nebo simulací virtuálních robotů, lze také využít [11]. Mezi největší výhody využití Lego Mindstorms ve výuce základů algoritmizace bezpochyby patří právě atraktivní interaktivnost učebního procesu, kdy studenti okamžitě po návrhu algoritmu vidí výslednou reakci robota a pokud jejich algoritmus nebyl zcela dokonalý, vidí i tím způsobené následné chyby, což přispívá k lepší efektivitě při závěrečném ladění algoritmu [12]. Popularita Lego Mindstorms je podpořena také nejrůznějšími soutěžemi, z nich neznámější je FIRST LEGO League, která je dotována přímo Ministerstvem školství, mládeže a tělovýchovy a to právě i proto, že soutěž má každý rok unikátní téma. Kupříkladu v roce 2012 soutěž nesla název „Senior Solutions“ a byla zaměřena na pomoc seniorům a tím pádem úkol soutěžících bylo navrhnout efektivní řešení daného problému, který by starým lidem usnadnil život [13].

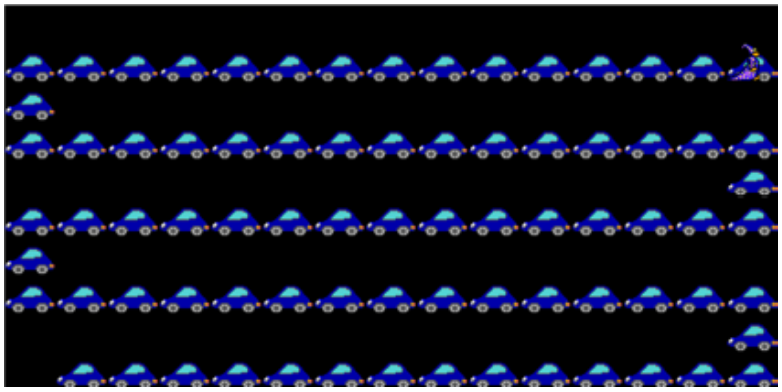
2.4 Baltík

Čaroděj Baltík elegantně spojuje klasické programování s moderním grafickým vzhledem a tím pádem nenáročnou formou zasvěcuje děti i dospělé do světa programování a rozvíjí jejich logické myšlení a tvořivost. Výukový programovací jazyk Baltík vychází z programovacího ná-

stroje Baltazar, který vznikl roku 1993. Pracoval na bázi programovacího jazyka C a byl vyvinut českou společností SGP Systems [14]. Baltík je založen na programování příkazů pomocí ikonek, čímž začátečníkům odpadá nutnost naučit se složitou textovou syntaxi. Pomocí ikon příkazů přikazujeme základní ovládání Baltíka (popojdi, doprava, čekej), základní programovací struktury (podmínka, cyklus) a příkazy pro práci s proměnnými. Existuje také možnost vytvářet procedury, které nalezneme pod příkazem s názvem pomocník, ale vytvářet pokročilé procedury s návratovou hodnotou zde nelze.

Ukázka pokročilých úloh:

Naprogramujte program KOLONA AUT, který bude ve výsledku odpovídat níže uvedenému obrázku [15].

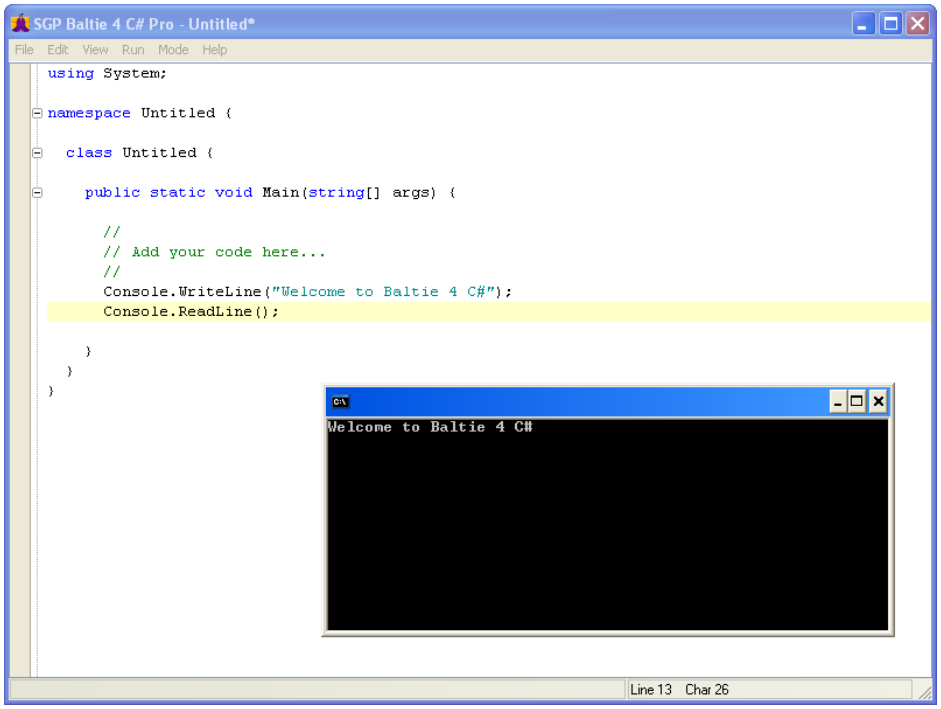


Obrázek 7 Baltík - Program KOLONA AUT

Při řešení je pro přehlednost a značnou časovou úsporu využit vnořený cyklus.

Obrázek 8 Baltík – Řešení KOLONA AUT

Po získání patřičných znalostí principů algoritmizace, lze navázat na verzi Baltie 4 C# Pro, která obsahuje plnohodnotný textový editor C#, pro lepší orientaci zde funguje zvýraznění syntaxe a obsaženy jsou také návrhy na doplňování kódu [16].



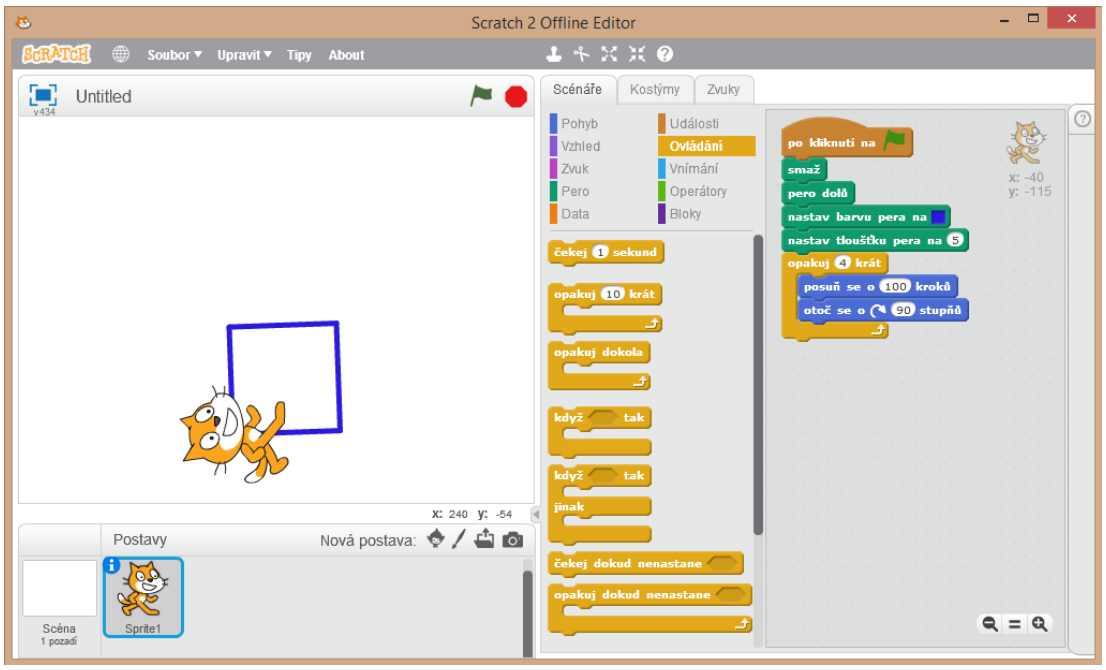
Obrázek 9 Prostředí SGP Baltie 4 C# PRO [17]

2.5 Scrach

Výukový programovací jazyk Scratch byl vyvinut skupinou pojmenovanou LifeLong Kindergarten, což ve volném překladu znamená *celý život ve školce*, a to konkrétně v Mediální laboratoři Massachusettského Institutu Technologií v roce 2007. Programovací jazyk Scratch a jeho vývojové prostředí je zcela intuitivní pro studenty, kteří nemají předcházející zkušenost s programováním, a to především přehledným uživatelským rozhraním. Nalezneme zde panel s nástroji, panel skriptů a scénu, kde lze názorně vidět plnění příkazů řízeným objektem zvaným sprite – skřítek. Panel nástrojů je rozdělen na osm skupin, které prošly empirickým testováním [18]. Výsledek testu ukázal, že usnadňují začátečníkům poznávání programování, přičemž první skupina obsahuje příkazy k ovlivnění pohybu – posun, otáčení. Následující tři skupiny umožňují nastavení vzhledu sprita, přehrávání zvuků nebo zanechávání názorných stop pohybu. Pátá skupina obsahuje příkazy pro řízení chodu programu – sekvence příkazů, podmínky (if, if-else) a cykly (while, for). Další slouží k vracení hodnot – aktuální čas, velikost hlasitosti apod. Poslední dvě sekce obsahují základní matematické funkce (sčítání, odčítání, porovnávání), možnost náhodně generovat čísla, práci s textovými řetězci a v neposlední řadě také vytváření seznamů a také jak globální, tak přímo lokální proměnné pro určitého sprita [18].

Ukázka úloh:

Použijte příkaz cyklu pro nakreslení čtverce.



Obrázek 10 Scratch - nakreslení čtverce

Výhodou je existence jak off-line editoru, který je v dispozici pro všechny platformy od Windows, Mac OS X přes Linux, tak také možnost využití on-line verze.

2.6 Kritéria hodnocení a výběr

Mezi hodnotící kritéria patří přívětivost prostředí, návaznost na vyšší programovací jazyk, typ licence, druh verze (on - line, off - line), algoritmické vlastnosti a v neposlední řadě také jazyková mutace – složitost jazykových znalostí.

Imagine Logo

Přívětivost prostředí: vizuálně povedený objektový programovací jazyk

Návaznost na vyšší programovací jazyk: je možné pokračovat s jakýmkoliv objektově orientovaným jazykem kupříkladu Java či Visual Basic

Licence: neomezená školní licence stojí 9 520 Kč, ale poté plnou verzi mohou studenti i učitelé bezplatně využívat z pohodlí domova

Druh verze: off - line

Algoritmické vlastnosti: základní matematické funkce, podmínka, cyklus, rekurze

Jazyk: česká lokalizace

Karel

Přívětivost prostředí: graficky zastaralé prostředí

Návaznost na vyšší programovací jazyk: Java

Licence: zdarma

Druh verze: off – line i on - line

Algoritmické vlastnosti: podmínka, cyklus, rekurze, celočíselné proměnné, vícerozměrná pole

Jazyk: česká lokalizace

Lego Mindstorms

Přívětivost prostředí: vynikající programovatelná robotická stavebnice

Návaznost na vyšší programovací jazyk: C

Licence: vývojové prostředí zdarma, ale je nutno zakoupit stavebnici LEGO Mindstorms EV3 (9300 Kč)

Druh verze: off - line

Algoritmické vlastnosti: matematické operace, příkazy, podmínky a cykly

Jazyk: česká lokalizace

Baltík

Přívětivost prostředí: vzor pro celou řadu dalších výukových programovacích nástrojů [16]

Návaznost na vyšší programovací jazyk: C#

Licence: pro celou školu: roční licence – 3500 Kč, časově neomezená licence 25 000 Kč

Druh verze: CD s off – line verzí

Algoritmické vlastnosti: podmínky, cykly, základní matematické funkce, příkazy pro práci s proměnnýma, soubory, adresáři, pouze procedury

Jazyk: česká lokalizace

Scratch

Přívětivost prostředí: intuitivní ovládání a přehledné uživatelské prostředí

Návaznost na vyšší programovací jazyk: žádnou přímou návaznost nemá, ale dokáže položit skvělý základ, na kterém se dá stavět

Licence: zdarma

Druh verze: off – line i on – line

Algoritmické vlastnosti: posloupnost příkazů, cykly, podmínky, proměnné a seznamy, procedury a funkce (novinka v on – line prostředí)

Jazyk: česká lokalizace

Výběr algoritmického nástroje pro tvorbu metodických listů byl ovlivněn výše uvedenými kritérii a také nainstalováním a poté otestováním pomocí základních algoritmických úloh viz předešlé kapitoly pojednávající o jednotlivých algoritmických nástrojích. Nejvíce kladněji působil výukový programovací jazyk Scratch, který byl následně pro tvorbu metodických listů vybrán. Na rozdíl od dalších algoritmických nástrojů se u něj skvěle kombinuje jeho velmi přívětivé české vývojové prostředí, které je současně on – line (formou SaaS) zdarma k dispozici a současně obsahuje nástroje vhodné k základní výuce algoritmizace, a to jak jednoduché příkazy, tak i pokročilé cykly while, for, také podmínky if, if – else a v neposlední řadě procedury a funkce.

II. PRAKTICKÁ ČÁST

3 METODICKÉ LISTY PRO VÝUKU PROGRAMOVÁNÍ

Vypracované metodické listy pro výuku programování spadají dle Rámcového vzdělávacího programu do vzdělávací oblasti Informační a komunikační technologie. Tvorba metodických listů byla ovlivněna informacemi získanými během studia didaktických předmětů, a to jak na Fakultě aplikované informatiky na Univerzitě Tomáše Bati ve Zlíně - zejména z předmětů Didaktika informatiky a dále Učení a vyučování [19], tak také znalostmi a dovednostmi z předchozího bakalářského studia na Pedagogické fakultě na Univerzitě Palackého v Olomouci. Následně byly tyto informace přeměněny v začátečnické zkušenosti prostřednictvím následné pedagogické praxe na Purkyňově gymnáziu ve Strážnice.

3.1 Metodický list 1

Název

Vývojového prostředí Scratch

Stručná anotace

Studenti se seznámí s vizuálním, výukovým programovacím jazykem Scratch a jeho online vývojovým prostředí, čímž získají orientaci ohledně možností využití dostupných programovacích nástrojů.

Časová dotace

45 minut

Vstupní znalosti

- základní znalost práce s počítačem

Pomůcky

- počítač s internetovým připojením a aktualizovaným internetovým prohlížečem

Kognitivní cíle

- student si osvojí pojem algoritmus
- student má poznat vývojové prostředí Scratch
- student se orientuje a následné využívá dostupné nástroje vývojového prostředí Scratch
- student je schopen vytvořit první funkční program pomocí posloupnosti příkazů

Klíčové slova

algoritmus, vývojové prostředí Scratch, panel nástrojů, posloupnost příkazů

Rozpracované téma

Výuku začnete tím, že studentům objasníte pojem algoritmus. Algoritmus je jednoznačný a přesný popis řešení problému, který stanovuje po sobě jdoucí kroky nutné pro realizaci požadované akce, pomocí kterých může i laik kupříkladu vyžehlit košili [20]. Zeptejte se studentů, zda někdy žehlili košili. Pokračujte dotazem, zda by dokázali vytvořit slovní zápis algoritmu (postupu) pro vyžehlení košile. Návrhy zaznamenejte na tabuli a poté je upravte do následné podoby.

Slovní zápis algoritmu – žehlení košile:

Formulace problému

Vyžehli košili.

Analýza úlohy

Vstupní údaje – žehlička, žehlící prkno, košile

Výstupní údaje – vyžehlená košile

Analýza – žehlete, dokud je košile zmačkaná

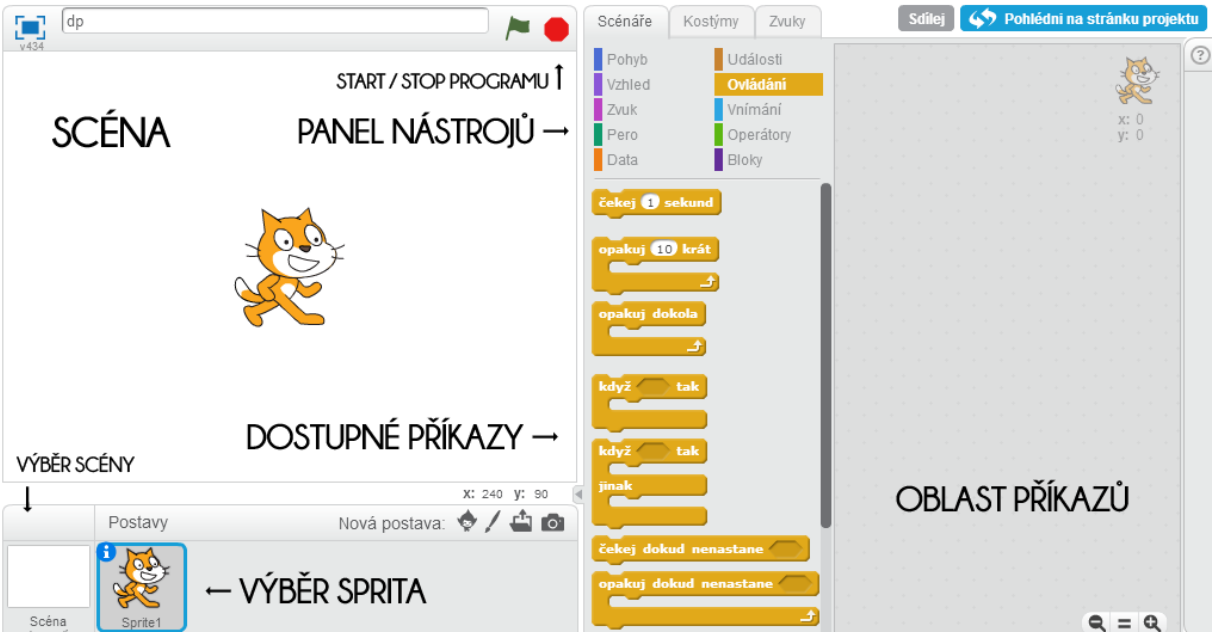
Sestavení algoritmu – slovní popis

1. Vezměte si žehlící prkno, žehličku a košili
2. Poté si položte košili na žehlící prkno.
3. Žehličku přiložte na košili a pohybujte jí.
4. Je košile zmačkaná?
   ANO – zopakujte bod číslo 3
   NE – pokračujte bodem číslo 5
5. Pověste košili na ramínko

Domácí úkol

Zadejte studentům domácí úkol, aby sestavili vlastní slovní zápis algoritmu. Dodejte, že algoritmus by měl řešit problém z každodenního života.

Poté studentům představte programovací jazyk Scratch a ním spjaté zcela intuitivní on – line vývojové prostředí (http://scratch.mit.edu/), kde návrh algoritmů probíhá.

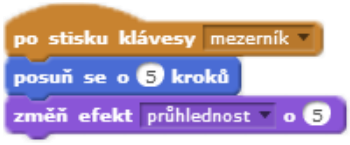


Obrázek 11 Popis prostředí Scratch 2.0

Za zmínku zcela jistě stojí možnost registrace a s tím spjatý přístup ke svým projektům z jakéhokoliv místa (možnost pro tvorbu závěrečných domácích prací). Dejte časový prostor pro tuto registraci. Následně studentům ukažte již existující poutavé projekty, které jsou dostupné na výše uvedeném serveru.

Cvičný příklad

Otevřete nový soubor, poté v panelu nástrojů vyberte scénář – události, který Vám nabídne dostupné příkazy. Příkazy se jednoduše skládají do sebe. Tvorbu prvního programu začtete přetažením příkazu **po stisku klávesy mezerník** do oblasti příkazů. Dále využijte scénář pohybu, který Vám nabídne příkaz **posuň se o 5 kroků**. Aby se sprite pouze nepohyboval, doplníme program o konečný příkaz ze scénáře vzhledů a to **změn efekt průhlednosti o 5**.



Obrázek 12 První program

Spuštění programu dojde pomocí klávesy mezerník, poté se tedy sprite posune o pět kroků a jeho průhlednost vzhledu se sníží o hodnotu pět. Po opakovaném spuštění se hodnota průhlednosti neustále snižuje, až sprite zcela zmizí.

Samostatná práce

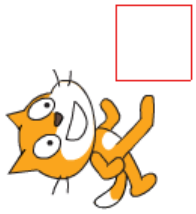
Samostatné práce obecně slouží k upevnění učiva a prostřednictvím praktické práce vedou k lepšímu zapamatování získaných poznatků. Důležitým prvkem je jasné a srozumitelné zadání pokynů. Nezapomeňte procházet mezi počítači a kontrolujte, jak studenti samostatnou práci vždy zvládají [21]. Následující samostatná práce ověřuje orientaci studentů ve vývojovém prostředí Scratch a také schopnost využívat dostupné nástroje pro následnou tvorbu programu.

Zadání

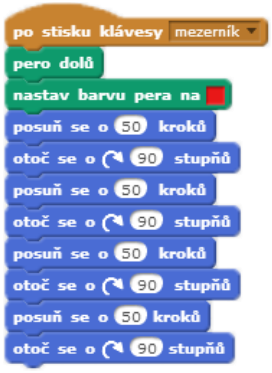
Vytvořte sled příkazů pomocí, kterých sprite nakreslí červený čtverec. Pro možnost pozdější dostupnosti, nezapomeňte projekt uložit.

Řešení

Program začnete některým ze zahajovacích příkazů (program lze také spustit pouhým kliknutím levým tlačítkem myši na vytvořený sled příkazů), poté nastavte směr kreslení pera a zvolte příkaz na změnu jeho barvy. Dále zvolte příkaz pro posun o vybraný počet kroků (tím dojde k nakreslení úsečky) a následujte ho příkazem **otoč se o 90 stupňů**. Po několika opakováních dojde k tíženému výsledku viz obrázek 13.



Obrázek 13 Samostatná práce1



Obrázek 14 Řešení samostatné práce 1

3.2 Metodický list 2

Název

Cykly

Stručná anotace

Studenti plynule navážou na získané zkušenosti z předchozího metodického listu a dále je zdokonalí o opakovaně prováděné posloupnosti příkazů – cykly.

Časová dotace

45 minut

Vstupní znalosti

- základní znalost práce s počítačem
- orientace ve vývojovém prostředí Scratch
- znalosti posloupnosti příkazů

Pomůcky

- počítač s internetovým připojením a aktualizovaným internetovým prohlížečem

Kognitivní cíle

- student chápe princip cyklu a efektivnost jeho využití

Klíčové slova

cyklus s určitým počtem opakování (REPEAT), cyklus s podmínkou (REPEAT – UNTIL), vývojové prostředí Scratch

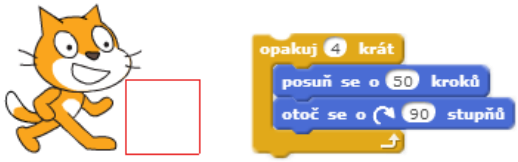
Rozpracované téma

Věnujte patřičný čas na kontrolu domácího úkolu spjatého se slovním zápisem algoritmu. Pokračujte tím, že připomenete studentům předchozí samostatnou práci, která požadovala vytvoření algoritmu, s jehož pomocí sprite nakreslí čtverec. Otevřete si tuto práci a chtějte, aby studenti analogicky vytvořili algoritmus na nakreslení dalších geometrických útvarů (obdélník, trojúhelník a jiné).

Obrázek 15 Trojúhelník

Cvičný příklad 1

Poté se opětovně vraťte k uložené původní verzi samostatné práce, kde naleznete posloupnost příkazů pro nakreslení čtverce. Zeptejte se studentů, zda někoho napadá, zda by šel tento program lépe zefektivnit. Posléze jim vysvětlete, že to lze a to použitím cyklu.



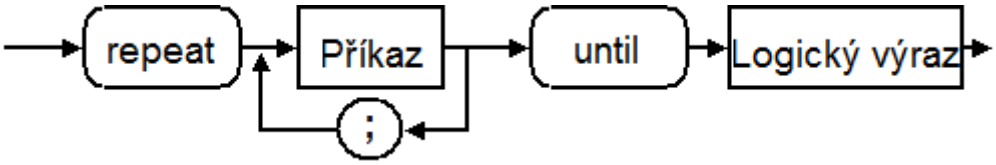
Obrázek 16 Cyklus

Domácí úkol

Ukažte studentům kresbu rozmanitých obrazců pomocí cyklu a zadejte jim na příští setkání úkol, aby zapřemýšleli nad jednoduchým cyklem s opakováním základních příkazů a poté experimentovali jeho hodnotami a tím vytvořili zajímavé kresby, které předvedou posléze svým spolužákům.

Obrázek 17 Cyklus obrazec

Studenti se tedy seznámili s cyklem REPEAT, který se po daném počtu opakování ukončí. Nyní je uveďte do problematiky ohledně cyklu REPEAT – UNTIL. Cyklus pracuje na principu, že vše, co je mezi příkazy repeat a until se opakuje do té doby, dokud není splněna zadaná podmínka.



Obrázek 18 Syntaxe cyklu REPEAT – UNTIL [20]

Cvičný příklad 2

Pro ukázku cyklu REPEAT – UNTIL, zvolte opět příklad pro nakreslení čtverce, jehož zadání upravte do podoby: Vytvořte sled příkazů pomocí, kterých sprite bude kreslit červený čtverec do doby, než stisknete klávesu s na znamení stopnutí programu.

Obrázek 19 Cyklus REPEAT – UNTIL

Samostatná práce

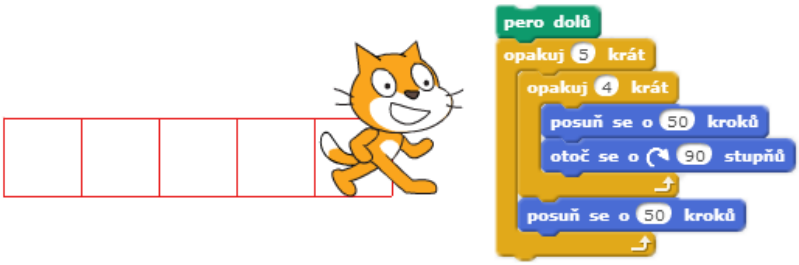
Samostatná práce slouží k ověření, zda studenti chápou pojem cyklus a dovedou tyto teoretické poznatky využít v praxi při řešení algoritmických úloh. Studenti tak své pomocí přijdou na princip vnořených cyklů.

Zadání

Vymyslete algoritmus, který nakreslí pět čtverců vedle sebe s využitím znalostí cyklů. Poté jej zapište ve vývojovém prostředí Scratch.

Řešení

Doporučte studentům využít cvičný příklad 1, kde sprite pomocí cyklu REPEAT nakreslil čtverec. Tento kód vložte do nově vytvořeného cyklu, který se bude pětkrát opakovat. Vše se ale musí doplnit o další příkaz, aby došlo opravdu k nakreslení čtverců vedle sebe, musí se sprite po nakreslení každého čtverce posunout o 50 kroků (délka strany čtverce).



Obrázek 20 Samostatná práce - vnořený cyklus

3.3 Metodický list 3

Název

Pokročilejší cykly a proměnné

Stručná anotace

Studenti budou schopni pracovat s proměnnými a i nadále budou prohlubovat předchozí nabyté znalosti o cyklech, jelikož je to jedna několika nejpodstatnějších struktur, které je nutno procvičovat.

Časová dotace

2x 45 minut

Vstupní znalosti

- základní znalost práce s počítačem
- schopnost práce s posloupností příkazů a cykly

Pomůcky

- počítač s internetovým připojením a aktualizovaným internetovým prohlížečem

Kognitivní cíle

- student chápe pojem proměnná a aktivně jej využívá
- student je schopen vytvořit algoritmus s využitím cyklů a proměnných

Klíčové slova

vývojové prostředí Scratch, algoritmus, cyklus REPEAT, cyklus REPEAT – UNTIL, proměnná

Rozpracované téma

Vývojové prostředí Scratch umožňuje využít nejen cyklus REPEAT s určitým počtem opakování anebo cyklus s podmínkou REPEAT – UNTIL, ale také tzv. nekonečný cyklus – označovaný jako FOREVER, v české lokalizaci Scratch jej naleznete pod příkazem **pořád dokola.** Vysvětlete studentům, že nekonečný cyklus nemá ani vstupní ani výstupní podmínku a proto jej nelze opustit [22].

Cvičný příklad 1

Představte studentům nekonečný cyklus, který bude obsahovat dva vnořené cykly, přičemž jeden bude sprita zmenšovat a další cyklus ho opět o stejnou hodnotu zvětší.

Obrázek 21 Nekonečný cyklus

Cvičný příklad 2

Připomeňte studentům cvičný příklad z předchozího metodického listu, který ukazuje, jak využít cyklus REPEAT při požadavku nakreslení čtverce. Pokuste se vytvořit program, ve kterém bude sprite v nekonečném cyklu kreslit čtverec. Příklad doplňte o vzhledové a zvukové prvky a vhodně zapojte příkaz **čekej**.

Obrázek 22 Nekonečný cyklus 2

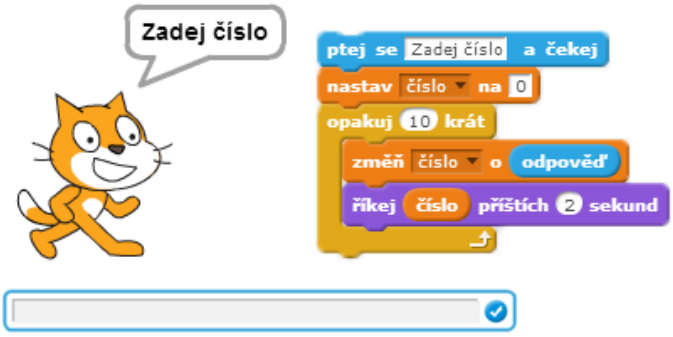
Nyní je načase studentům vysvětlit, co jsou to proměnné, jak se vytvářejí, k čemu slouží, a jak je používat v projektech. Proměnná může být globální tzv. použitelná celý projekt nebo lokální - lze ji využít pouze pro určitý sprite. Nejčastěji se proměnné využívají k ukládání hodnot. Na rozdíl od algebraické proměnné (které jsou obvykle neznámé), proměnné v Scratch obsahují vždy známé hodnoty. Proměnné mohou obsahovat textové řetězce, čísla nebo logické hodnoty - booleans (ano / ne). Příklady proměnných:

- obsahuje text
- 321
- 2.56
- 0
- - 256
- pravda
- {Nic - prázdný řetězec}

Proměnnou vytvořte pomocí scénáře data, kde naleznete tlačítko *vytvořit proměnnou*. Zadejte název proměnné, kterou chcete vytvořit a zvolte, zda bude globální nebo lokální.

Cvičný příklad 3

Následující příklad slouží k počáteční práci s proměnnými. Zadejte kupříkladu, že chcete, aby sprite dokázal vyjmenovat prvních deset násobků zvoleného čísla. Algoritmus bude následující: Nejprve vytvořte proměnnou s název číslo, jelikož to nebude žádný rozsáhlý projekt, je jen na Vás, zdali zvolíte její rozsah globálně či lokálně. Program začnete tím, že ze scénáře zvolíte nástroj vnímaní a poté příkaz **ptej se a čekej**, který bude sloužit k zvolení čísla, jehož násobky si přejete znát. Následuje výběr příkaz, který proměnnou číslo nastaví na výchozí hodnotu 0. Nejdůležitější je vytvořit poté cyklus deseti opakování, který ve svém těle obsahuje příkaz, kterým proměnná nabývá na své hodnotě.



Obrázek 23 Práce s proměnnou

Cvičný příklad 4

V pořadí čtvrtý cvičný příklad představuje možnost dalšího využití proměnné. Princip programu spočívá v tom, že do proměnné se bude ukládat počet již provedených opakování, jedná se o tzv. počítadlo. Na začátku jej standardně nastavte na hodnotu 1 a při každém zopakování cyklu se hodnota této proměnné navýší právě o 1, to znamená, že při druhém opakování bude mít hodnotu 2, při třetím hodnotu 3 apod. Je nutno pracovat i s jednoduchým operátorem, který přiřadí koncovou hodnotu, kterou má počítadlo nabýt a tím ukončí cyklus REPEAT – UNTIL.

Obrázek 24 Počítadlo

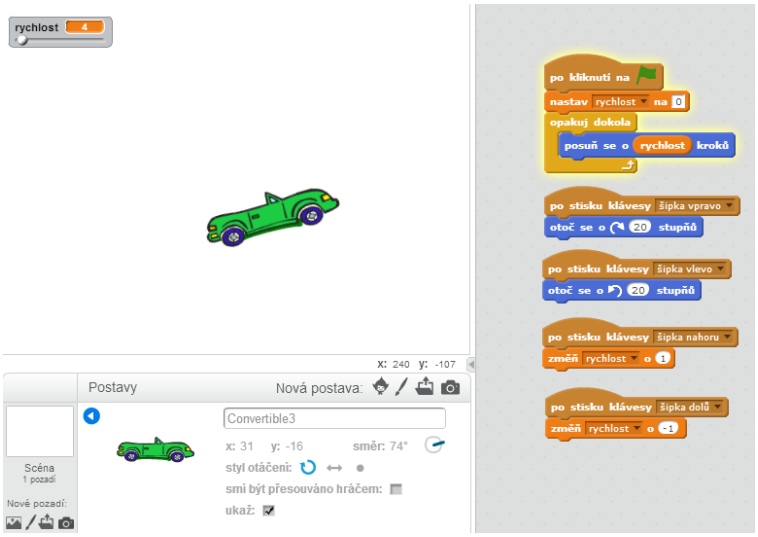
Domácí úkol

Zadejte studentům za úkol vytvořit program, který bude pracovat s počítadlem. Při spuštění se sprite dotáže na počet opakování a po zadání čísla provede patřičný počet opakování cyklu. Vše ostatní ponechejte na kreativitě studentů.

Obrázek 25 Počítadlo – domácí úkol

Samostatná práce

Nahrajte nového sprita – dopravní prostředek. Poté pomocí jednoduché práce s proměnnou vytvořte program, který bude sloužit k ovládání směru a rychlosti pohybu zvoleného dopravního prostředku.



Obrázek 26 Samostatná práce – proměnná

Doplňující příklad

Příklad zařaďte v případě dostatečné časové rezervy, jelikož nerozšiřuje algoritmické znalosti, ale pouze představuje širší využití proměnných specifické pro prostředí Scratch.

Vývojové prostředí Scratch, přesněji jeho scénář vnímání, obsahuje příkaz **aktuální**, jenž vrací aktuální hodnotu času. Z nabídky lze vybrat položku rok, měsíc, den, den v týdnu, hodinu, minutu a v neposlední řadě také sekundu. Chcete-li zobrazit hodnotu aktuálního času, klepněte na zaškrtávací políčko vedle příkazu. Vašim cílem je vytvořit scénu, která bude zobrazovat všechny aktuální časové položky.

Obrázek 27Kalendář

Algoritmus bude využívat nekonečného cyklu, který bude obsahovat proměnou, která se bude nastavovat pomocí příkazu aktuální čas. Po každou časovou položku je nutné vytvořit nového sprita a také novou proměnnou.

Obrázek 28 Zdrojový kód - kalendář

3.4 Metodický list 4

Název

Operátory

Stručná anotace

Studenti budou schopni pracovat se základními matematickými funkcemi.

Časová dotace

45 minut

Vstupní znalosti

- základní znalost práce s počítačem
- schopnost práce s posloupností příkazů a cykly
- dovednosti řešení algoritmů

Pomůcky

- počítač s internetovým připojením a aktualizovaným internetovým prohlížečem

Kognitivní cíle

- získání schopnosti správně se orientovat v možnostech využití operátorů
- osvojení pokročilých znalostí proměnných
- student je schopen vytvořit algoritmus s využitím operátorů

Klíčové slova

vývojové prostředí Scratch, algoritmus, proměnná, operátory, výroková logika

Rozpracované téma

Operátory jsou prvky, které slouží k práci se základními matematickými funkcemi (sčítání, odčítání a jiné), mohou generovat náhodná čísla, porovnávají větší – menší atd. Obsahují také výrokovou logiku (and, or, not) a dokáží pracovat s textovými řetězci - spojovat je, vracet jejich délku [23].

Vývojové prostředí momentálně nabízí kromě jedenácti základních operátorů, také možnost využít šest logických operátorů.

Základní operátory

Obrázek 29 Základní operátory

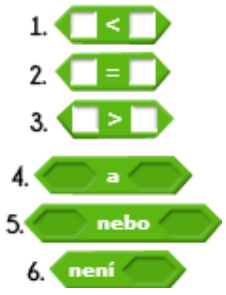
1. Blok sčítá dvě zadané hodnoty a vrací výsledek.
2. Blok odečítá druhou hodnotu od první a vrací výsledek.
3. Blok násobí dvě hodnoty a vrací výsledek.
4. Blok dělí první hodnotu, druhou a vrací výsledek. Je-li první hodnota nedělitelná druhou, výsledek bude obsahovat desetinná místa.
5. Blok vybere náhodné číslo v rozmezí od první zadané hodnoty po druhou, včetně obou koncových hodnot. Pokud se obě hodnoty neobsahují desetinná místa, výsledek bude také celé číslo. Kupříkladu, pokud zadáme do první hodnoty 1 a druhou hodnotou zvolíme 3, blok vrátí výsledek 1, 2 nebo 3. Pokud jedna ze zadaných hodnot obsahuje desetinnou čárku, vrací se hodnota také s desetinným číslem. Například, pokud jsou k dispozici hodnoty 0,1 a 0,14, výstup bude 0,1, 0,11, 0,12, 0,13 nebo 0,14.
6. Blok slouží ke spojování čísel, slov, vět do jedné výstupní hodnoty viz obrázek 30.

Obrázek 30 Blok spoj (join)

7. Používá se zejména v případech, kdy chcete najít písmeno uprostřed řetězce textu anebo chcete-li zjistit, zda dané písmeno v řetězci rovná určité hodnotě.
8. Blok vrací ve výsledku délku zadané hodnoty.
9. Využívá se při dělení hodnot, přičemž pracuje se zbytek po dělení.
10. Blok zaokrouhlí zadanou hodnotu na nejbližší celé číslo. Platí zde standardní pravidla pro zaokrouhlování; desetinná čísla, které jsou 0,5 nebo vyšší se zaokrouhlí nahoru, zatímco desetinných čísla pod 0,5 se zaokrouhlí dolů.
11. Aplikuje zvolenou funkci na zadanou hodnotu a vrací výsledek. Dovede pracovat s absolutní hodnotou, druhou mocninou a odmocninou, goniometrickými funkcemi, exponenciální funkcemi, Eulerovým číslem a dalšími.

Logické operátory

Patří k důležitým prvkům, které se používají v podmínkách ke spojení několika částí podmínky. Slouží kupříkladu k zjištění, zda je proměnná **P** v rozmezí od 1 do 10. K dosažení výsledku použijte dvě porovnání a mezi ně dejte logický operátor **A.**



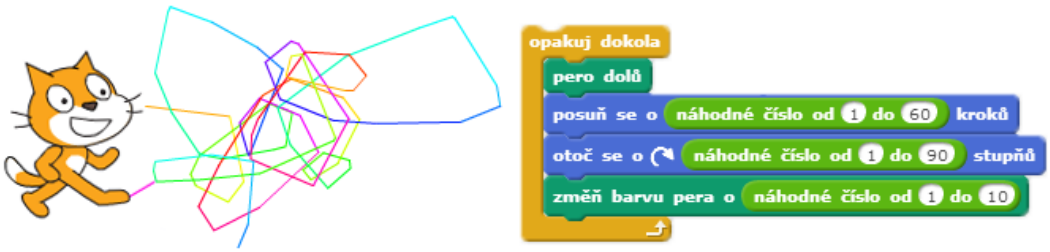
Obrázek 31 Logické operátory

1. Operátor kontroluje, zda první hodnota je menší než druhá hodnota. Je-li menší, vrací hodnotu pravda (true); pokud ne, vrátí nepravdu (false).
2. Srovnává, zda je první hodnota rovna jiné hodnotě. Pokud hodnoty jsou stejné, vrátí blok pravdu (true); pokud ne, vrátí nepravdu (false).

3. Porovnává, zda první hodnota je větší než druhá. Je-li menší, vrací blok pravdu (true); pokud ne, vrátí nepravdu (false).
4. Spojuje dva logické bloky. Jestliže jsou obě dvě hodnoty pravdivé, vrací hodnotu pravda (true). Pokud není pravdivá jedna hodnota nebo zároveň obě dvě, vrací hodnotu nepravda (false).
5. Blok spojuje dva logické bloky, takže pokud alespoň jeden z nich je pravdivý, vrací blok hodnotu pravda (true). Pokud ani jedna hodnota není pravdivá, vrátí hodnotu nepravda (false).
6. Logický operátor, který slouží k negaci. Kupříkladu použijete-li logický operátor **není** před podmínkou, výsledná podmínka bude splněna pouze v tom případě, když původní podmínka splněna nebude.

Cvičný příklad

S využitím základních operátorů vytvořte program, jehož pomocí bude sprite náhodně kreslit. Generování čísel využijte, jak pro délku přímky, tak pro její sklon a také pro změnu barvy pera.



Obrázek 32 Generování čísel

Samostatná práce

Sdělte studentům, aby opětovně otevřeli cvičný příklad 3 z předchozí hodiny a popřemýšleli, jak vylepšit kód tak, aby sprite po vyjmenování celé násobilky daného čísla, ještě pro přehlednost celou násobilku vypsal. Tip na závěr: Je nutno použít další proměnnou!

Obrázek 33 Samostatná práce - násobilka

3.5 Metodický list 5

Název

První hra – Střílení slepic

Stručná anotace

Studenti budou schopni naprogramovat svou první hru, která jistě v jejich očích ještě zatraktivní programovací jazyk Scratch. Hra Střílení slepic vytvořená na náměty oblíbené kultovní střílečky Moorhunh pracuje s nekonečným cyklem, cyklem REPEAT – UNTIL, využívá také náhodného generování čísel a v neposlední řadě dochází k aplikaci proměnných (skóre, čas).

Časová dotace

2 x 45 minut

Vstupní znalosti

- základní znalost práce s počítačem
- schopnost práce s posloupností příkazů a cykly
- aplikace operátorů a proměnných

Pomůcky

- počítač s internetovým připojením a aktualizovaným internetovým prohlížečem

Kognitivní cíle

- student aktivně aplikuje dosud získané poznatky při tvorbě hry
- efektivní práce s proměnnýma, náhodným generováním čísel
- osvojení principu klonování objektů
- přemýšlení nad dalšími možnostmi rozšíření hry

Klíčové slova

vývojové prostředí Scratch, algoritmus, cyklus REPEAT, cyklus REPEAT – UNTIL, proměnná, generování čísel

Rozpracované téma

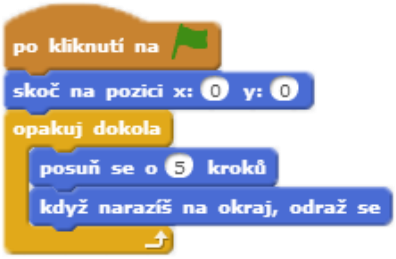
Zpracování jednoduché hry testuje míru nabytých znalostí získaných z předcházejících metodických listů, které měli studenty naučit ovládat problematiku základních algoritmických prvků – cyklů, proměnných a operátorů.

Začnete tím, že si založíte nový projekt a poté odstraňte dosavadního skřítka. Jelikož je Vašim cílem vytvořit hru na náměty kultovní střílečky Moorhunh, je zcela na místě použít pro nového sprita vzhled této legendární slepice. Pokud chcete studenty zapojit i po grafické stránce a máte dostupný vhodný software, nechejte stažení obrázku a následnou úpravu (ořez) výhradně na nich. Upozorněte je na výsledný grafický formát souboru. Prostředí Scratch totiž pracuje s více grafickými formáty, ale pouze u obrázku uloženém v grafickém formátu PNG, který na rozdíl od jiných formátů podporuje průhlednost, docílíte toho, že sprite bude obsahovat pouze ořez bez jakýchkoliv bílých okrajů [24]. Následně zvolte adekvátní pozadí.

Obrázek 34 Moorhunh ve Scratch

1) Nyní promyslete algoritmus, který po spuštění spritu přikáže nekonečný pohyb a následně ošetří to, aby se sprite nedostal mimo okraj scény.

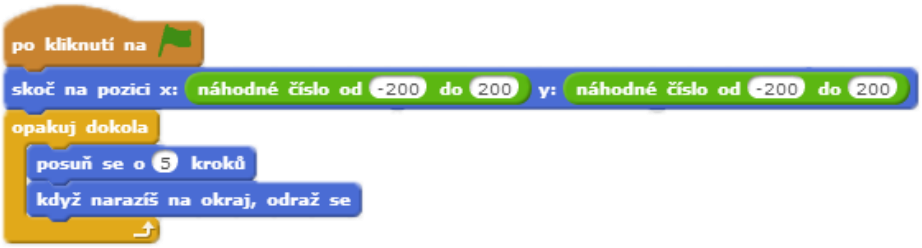
Řešení: Nastavte počáteční souřadnice, kde se sprite objeví. Dále vytvořte nekonečný cyklus, který bude obsahovat příkaz **posun se o 5 kroků** (ovlivňuje rychlost pohybu, pokud tedy chce ztížit obtížnost, nastavte větší počet kroků) a také další pohybový příkaz **když narazíš na okraj, odraž se.** Tip: Zastavte neustále rotování sprita. Klikněte na informační ikonu a poté na tečku, která tento pohyb zastaví.



Obrázek 35 ScratchMoorhunh - první část

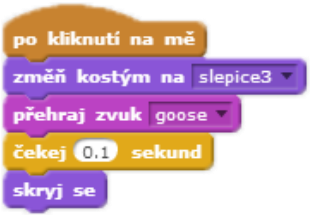
2) Kód pro neustálý pohyb sprita máte. Zvyšte obtížnost hry - sprite se objeví pokaždé na jiném místě.

Řešení: Stačí, aby v dosavadním algoritmu přidali do příkazu (**skoč na pozici x: 0 y:0**) operátor, který náhodně generuje čísla. Čím větší rozpětí hodnot, tím větší náhodnost pohybu.



Obrázek 36 ScratchMoorhunh - druhá část

Opětovně je potřeba nahrát další vzhled sprita, který se objeví při kliknutí (střelbě) na slepici. Postupujte podle předchozího návodu pro vytvoření sprita. Poté doplňte kód, hned za příkaz spuštění, o příkaz **změn kostým na první obrázek**, čímž docílíte toho, že se po spuštění hry načte původní sprite a ne obrázek střelené slepice. Další sled příkazů bude mít za úkol při kliknutí na slepici následně zobrazit obrázek střelené slepice. Vše doplňte vhodným zvukem a příkazem čekej. V neposlední řadě je důležité doplnit dva příkazy a to příkaz **skryj se**, který sprita po úspěšné střelbě skryje a také příkaz **ukaž se**, který sprita při opětovném spuštění hry zobrazí a ten doplňte ihned za počáteční příkaz pro spuštění hry.

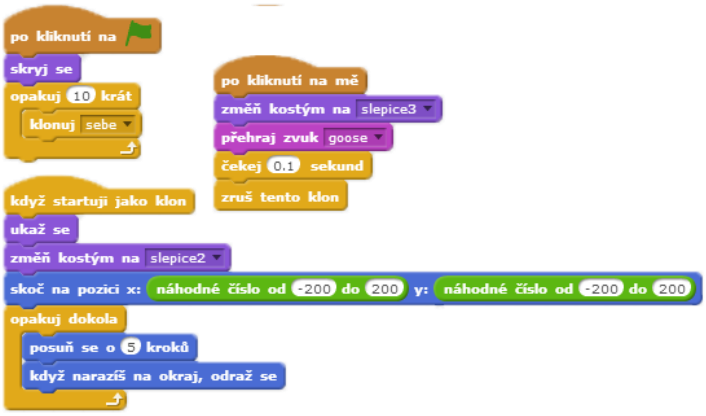


Obrázek 37 ScratchMoorhunh – algorimus střelby

3) Na řadu přichází ztížení obtížnosti hry a to způsobem, který je zcela specifický pro vývojové prostředí Scratch – klonováním. Každá naklonovaná slepice se tak stane instancí

původního objektu slepice, to znamená, že má stejné vlastnosti, ale každá funguje zcela nezávisle na ostatních.

Řešení: Využijte klonovací příkaz **když startuji jako klon**, který zaměňte za příkaz **po kliknutí na vlajku** (spouštěcí příkaz). Tato část přikáže každé nově vytvořené instanci slepice její náhodně generovaný směr pohybu. Poté za spouštěcí příkaz připojte kód, který pomocí cyklu REPEAT vytváří daný počet klonů. Váš dosavadní kód by měl vypadat následovně:



Obrázek 38 ScratchMoorhunh - klonování

Otestujte dosavadní část hry! Při spuštění hry se hlavní sprite schová a zobrazí se pouze jeho klony (instance). Jednotlivé klony se samostatně spustí a každý z nich se bude náhodně odrážet po hrací ploše.

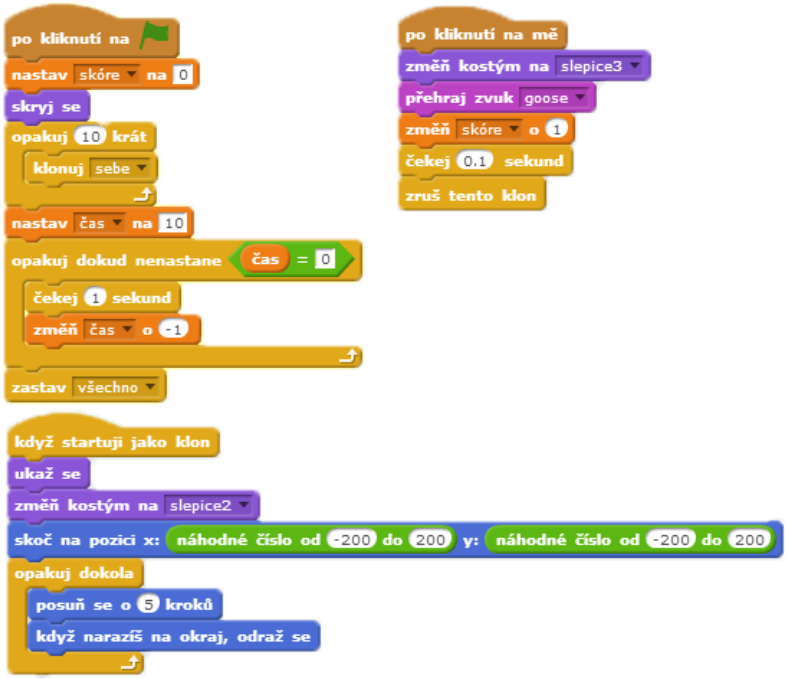
4) Udělejte hru ještě zajímavější – přidejte skóre a odpočítávání času.

Řešení: Nyní je zcela na místě vytvořit proměnnou, která bude sloužit ukládání dat ze skóre. Vytvořte tedy proměnnou s názvem skóre a její působnost zvolte globální, jelikož bude zaznamenávat hodnoty skóre všech spritů. Při spuštění hry přidejte příkaz, který nastaví hodnotu proměnné skóre na 0. Do druhé části příkazů, která ovlivňuje chování při kliknutí na sprita, vložte příkaz, jenž mění hodnotu proměnné o hodnotu 1. Otestujte funkčnost tím, že hru spustíte a při kliknutí (střelbě) na sprita se skóre navýší o hodnotu 1.

Použijte další proměnnou pro ukládání zbývajícího času. Pro přehlednost dejte nové proměnné název čas. Algoritmus odpočítávání času by měl být následovný:

- odpočítávání začne na deseti sekundách
- čas se bude odpočítávat po jedné sekundě
- hra skončí, když se čas dostane na nulu

Otestujte svou první hru. Pokud je příliš lehká, můžete přidat více slepic, zrychlit jejich pohyb, anebo snížit odpočítávání času.



Obrázek 39 ScratchMoorhunh

Tip: Pokud máte zbývající čas, zkuste hru nějakým zajímavým způsobem vylepšit. Kupříkladu jedna z řady možností je vytvoření dalších objektů (spritů), které není vhodné střílet a proto při každém zásahu tohoto objektu se hodnota skóre sníží, anebo objektů za jejich sestřelení hráč obdrží dvojnásobek bodů.

Obrázek 40 ScratchMoorhunh - rozšíření výsledné hry

3.6 Metodický list 6

Název

Podmínky

Stručná anotace

Studenti budou schopni vytvořit větvení programu na základě vyhodnocení zadané podmínky.

Časová dotace

2 x 45 minut

Vstupní znalosti

- základní znalost práce s počítačem
- schopnost práce s posloupností příkazů a cykly
- ovládání operátorů a proměnných

Pomůcky

- počítač s internetovým připojením a aktualizovaným internetovým prohlížečem

Kognitivní cíle

- student chápe pojem podmínka a aktivně jej využívá
- student je schopen vytvořit algoritmus, jak s využitím neúplného, tak s úplného podmíněného příkazu
- student ovládá podmínky ve spojení s logickými operátory

Klíčové slova

vývojové prostředí Scratch, algoritmus, podmínka, neúplný podmíněný příkaz, úplný podmíněný příkaz

Rozpracované téma

Vývojové prostředí Scratch nabízí dva druhy příkazových bloků, které lze využít při větvení programu na základě vyhodnocení zadané podmínky. První z nich lze zařadit mezi neúplné

podmíněné příkazy a je analogií klasického příkazu IF-THEN. Princip spočívá v tom, že příkazy umístěné uvnitř toho příkazového bloku se uskuteční jen za předpokladu, že je zadaná podmínka pravdivá.

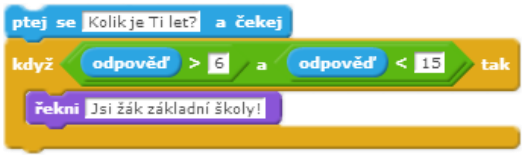
Cvičný příklad 1

Pomocí podmíněného příkazu **když tak,** zpracujte algoritmus, který bude cyklicky sprita posunovat až do té doby, kdy se dotkne okraje scény a poté až se tato podmínka splní, sprite řekne, že už nemůže dále a celý program se zastaví.

Obrázek 41 Podmínka IF-ELSE

Takové využití výše uvedeného neúplného podmíněného příkazu je specifické právě pro vývojové prostředí Scratch. Zdůrazněte proto, že nejčastěji se neúplný podmíněný příkaz využívá ve spojení s logickými operátory, a to k tvorbě kontrolních podmínek. Potřebujeme-li podmínku složitější, pak je ji možno složit z více podmínek pomocí právě logických (booleen) operátorů AND, OR či NOT.

- Neúplný podmíněný příkaz a logický operátor AND (a)



Obrázek 42 IF-ELSE a logický operátor AND

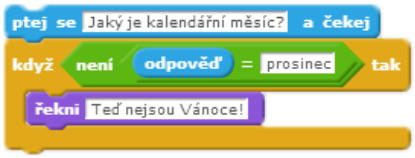
Pokud spojíme podmínky operátorem AND (a), pak je výsledná podmínka platná pouze v případě, že platí první i druhá část. Na příkladu se tedy výsledek „Jsi žák základní školy!“ se vypíše pouze v případě, že zadaný věk splňuje první část podmínky (věk je vyšší než šest let) a zároveň splňuje i druhou část podmínky (věk je nižší než patnáct let).

- Neúplný podmíněný příkaz a logický operátor OR (nebo)

Obrázek 43 IF-ELSE a logický operátor OR

V případě, že spojíme dvě podmínky logickým operátorem OR (nebo), pak je výsledná podmínka platná, pokud alespoň jedna z podmínek je pravdivá tzv. není platná v případě, že neplatí žádná ze zadaných podmínek. Výsledek „Super! Užívej si letní prázdniny.“ se vypíše, pokud při dotazu na kalendářní měsíc zadáte jeden nebo druhý z prázdninových měsíců.

- Neúplný podmíněný příkaz a logický operátor NOT (není)



Obrázek 44 IF-ELSE a logický operátor NOT

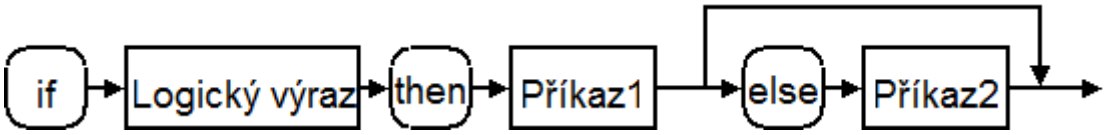
Výsledek je pravdivý, pokud podmínka není splněna (je nepravdivá). Výpis výsledku proběhne pouze v situaci, když zadáte jakoukoliv odpověď kromě prosince.

Domácí úkol

Požadujte po studentech vytvoření dalších tří algoritmů, které budou analogií výše uvedených příkladů. Pracují s neúplným podmíněným příkazem IF-THEN a zároveň využívají logických operátorů k porovnávání hodnot.

Cvičný příklad 2

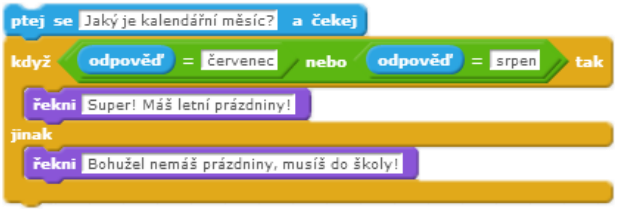
Druhý typ podmínky, úplný podmíněný příkaz, umožňuje specifikovat obě větve programu, a to jak větev, která se provede při splnění podmínky, tak také tu část, jejíž obsah je proveden při nesplnění podmínky. Analogicky odpovídá příkazu IF-THEN-ELSE, který můžete nalézt u vyšších programovacích jazyků.



Obrázek 45 Úplný podmíněný příkaz – syntaxe [20]

Vhodně upravte příklad, který se zabývá neúplným podmíněným příkazem s logickým operátorem OR (nebo) tak, aby nyní využíval úplného podmíněného příkazu. To znamená, že stačí

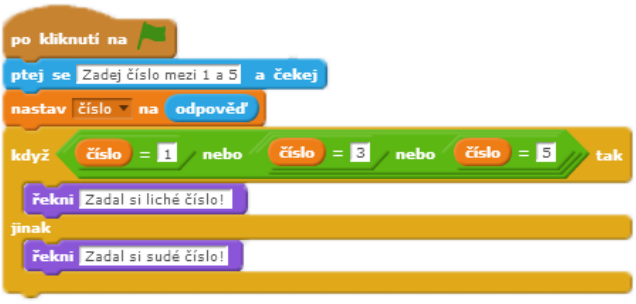
tento příklad doplnit o skutečnost, která se stane, pokud alespoň jedna z podmínek nebude splněna.



Obrázek 46 IF-THEN-ELSE

Cvičný příklad 3

Napište program, který požaduje zadání vstupní hodnoty kupříkladu číslo od 1 do 5, a poté s využitím úplného podmíněného příkazu určí, zdali jste zadali liché či suché číslo.



Obrázek 47 Lichost a sudost čísel

Tip: Upravte tento algoritmus tak, aby sprite zobrazil níže uvedený výsledek. Je nutno použít vhodný operátor.

Obrázek 48 Lichost, sudost čísel a operátor JOIN

Cvičný příklad 4

Vytvořte program, který bude sloužit jako jednoduché malování. Dle časové rezervy stanovte podmínky, které by měl daný program splňovat.

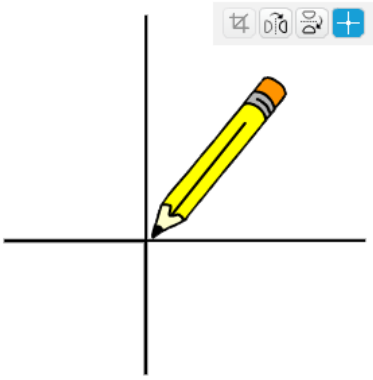
- Vytvoření kreslící tužky (15 minut)
- Přidání možnosti změny tloušťky tužky (10 minut)

- Možnost volby barevnosti tužky ovládaný stiskem vybrané klávesy (30 minut)

Vytvoření kreslící tužky

Po spuštění nového projektu odstraňte původního sprita a následně nahrajte nového se vzhledem žluté tužky, jenž je dostupný v knihovně prostředí Scratch. Základní algoritmus využívá ke kreslení pohyb myši, který bude následován kreslícím příkazem **pero.** Je nutné zde vytvořit nekonečný cyklus, který bude tyto příkazy, včetně příkazu **skoč na ukazatel myši,** neustále opakovat. Vše bude kontrolovat podmínka, která bude obsahovat příkaz **myš stisknuta** a pokud bude splněna, dojde provádění příkazu **pero dolů**, tedy ke kreslení na scéně. Pokud podmínka nebude splněna, provede se příkaz **pero nahoru** tzv. pero se zvedne ze scény a nebude docházet ke kresbě. Vyzkoušejte tuto část kódu kliknutím na spouštěcí vlajku a pak pohybem myší po scéně. Funguje to dle očekávání?

Tip: K nedokonalosti přispívá zcela jistě fakt, že ukazatel myši míří vždy do středu sprita (tužky) a tím pádem tužka nekreslí svým hrotem, ale od svého středu. Přenastavte proto tzv. střed kostýmu, který nalezte ve stejnojmenné záložce.



Obrázek 49 Nastavení středu tužky

Podle dostupné časové rezervy postupujte dále anebo přejděte na výuku dalšího metodického listu.

Změna tloušťky tužky

Za prvé, přidejte novou proměnnou s názvem "tloušťka čáry". Pokud si nejste jisti, jak ji vytvořit, vraťte se k metodickému listu číslo tři, který se zabývá proměnnými. Poté vložte příkaz, který po spuštění programu vždy nastaví tloušťku na výchozí hodnotu 1. Do nekonečného cyklu zařaďte příkaz **nastav tloušťku pera na** a na místo předem určené hodnoty dosaďte název proměnné tzv. tloušťka čáry tužky se poté bude dle Vašeho vlastního nastavení. Při vytvoření

proměnné se její hodnota zobrazuje na scéně, pro tento program bude zcela jistě nejlepší nastavit toto zobrazení na posuvník, kterým jde jednoduše měnit hodnota proměnné, tedy tloušťka tužky. Využijte výhodu posuvníku a to tak, že nastavíte minimální i maximální hodnotu proměnné – tloušťku pera poté nelze nastavit kupříkladu na hodnotu 90, která je příliš široká na kreslení obrazců. Nyní je čas pro ladění programu! Jistě jste si všimli, že tužka může kreslit kdekoliv na scéně, včetně posuvníku pro volbu proměnné. Zjistěte, na jakých souřadnicích tento posuvník nachází, a poté přidáním operátorů do podmínky tento problém vyřešte.

Obrázek 50 Tloušťka tužky s využití proměnné

Barevnost tužek

Přidejte dalšího sprita, využijte opět žlutou tužku a tu následně obarvěte nazeleno. Nezapomeňte nastavit střed opět na hrot tužky. Také je nutné nahrát další dva sprity (kaňky), které budou sloužit k výběru barvy. Zvolte dostupné obrázky a opětovně je přebarvěte. Nyní předveďte studentům specifický prvek, a to možnost zasílání zpráv ve spojení s následnou reakcí objektů. Po kliknutí na ikonu zelené barvy, je nutné vysílat zprávu pro zelenou tužku, která jí říká, aby změnila svůj kostým (přeměnila se na zelenou tužku) a zvolila odpovídající barvu pera. Začnete tedy tím, že přidejte zelené kaňce příkazy **po kliknutí na mě, rozešli všem zprávu zelená**. Poté klikněte na zelenou tužku a přidejte příkazy, které ho budou informovat, co má dělat, když obdrží zprávu zelená.

Obrázek 51 Reakce na obdrženou zprávu

Tip: Upravte kód tak, aby se každá barva přepnula pouze po stisku klávesy s jejím počátečním písmenem.

Domácí úkol

Někdy se stane, že nakreslíte něco, co zcela neodpovídá Vašim představám, proto do svého programu přidejte mazací gumu! Popř. také více barevných tužek. Postupujte obdobně jako při vytváření zelené tužky tzv. je nutno využít posílání zpráv mezi jednotlivými objekty.

3.7 Metodický list 7

Název

Závěrečný projekt – Hra s čísly

Stručná anotace

Závěrečný projekt má za cíl otestovat nabyté znalosti z předchozích metodických listů (cykly, proměnné, operátory, podmínky)

Časová dotace

45 minut

Vstupní znalosti

- základní znalost práce s počítačem
- schopnost práce s posloupností příkazů, cykly a podmínkami
- efektivní využití operátorů a proměnných

Pomůcky

- počítač s internetovým připojením a aktualizovaným internetovým prohlížečem

Kognitivní cíle

- student pracuje samostatně na závěrečném projektu
- je schopen vymyslet algoritmus a poté jej naprogramovat
- dovede vytvářet kreativní grafické pozadí projektu

Klíčové slova

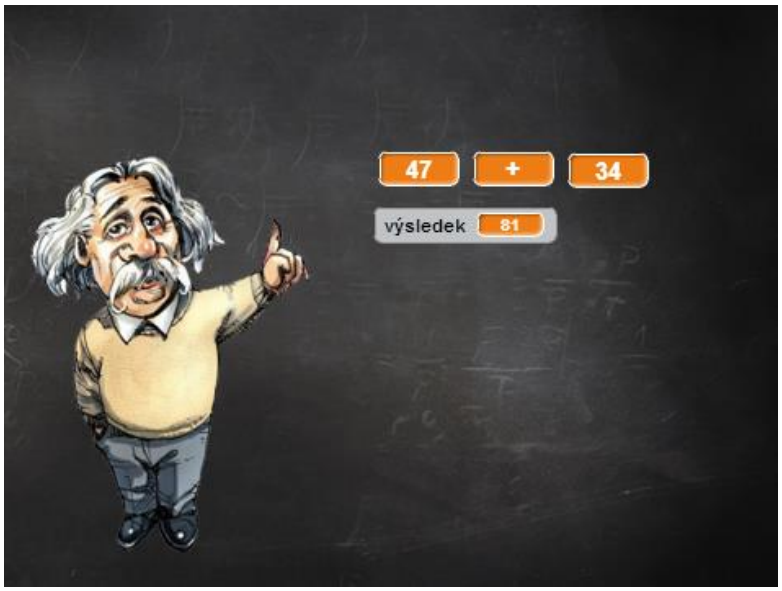
vývojové prostředí Scratch, algoritmus, samostatná práce

Rozpracované téma

Scratch kalkulačka

Již v předchozích metodických listech měli studenti za úkol vymyslet algoritmus programu, který by počítal násobky daného čísla a ty následně vypsal. V té době studenti nebyli ještě seznámeni s podmínkami, takže nebylo zcela na místě pouštět se do složitějších početních programů. Nyní na to přišel správný čas. Zadání je následovné:

Studenti mají za úkol vytvořit samostatně závěrečný projekt, který bude sloužit jako jednoduchá kalkulačka. Bude možno provádět základní početní operace jako je sčítání, odčítání, násobení a dělení. Samostatný projekt bude využívat všechny studentům známé algoritmizační struktury, jako jsou proměnné, podmínky, operátory a v neposlední řadě také specifické prvky pro vývojové prostředí Scratch – scénáře vnímání, vzhledu apod. Konečné hodnocení projektu bude postaveno na hodnocení jak funkčnosti projektu, tak také na jeho vizuální úpravě.



Obrázek 52 Ukázka kalkulačky

Algoritmus celého projektu může být následující: Po spuštění programu je sprite se Vás dotáže „Jaké početní operace chceš provádět?“ a pro názornost zobrazí dostupná znaménka a poté vy jedno zadáte. Každá početní operace bude obsahovat vlastní podmínku, která se provede pouze v případě, že odpověď se bude rovnat zadanému znaménku. Pokud ne, zkontrolují se postupně další tři podmínky a pokud se narazí na tu pravdivou, tak se provede. Princip spočívá dále v tom, že zadáte první číslo, které se posléze načte do proměnné a. Stejný postup se provede i zadání druhé hodnoty, která se tedy opět načte do proměnné, v tomto případě do proměnné b. Dále k samotnému výpočtu využijte početní operátory. Pro lepší vizuální efekt je nutno hodnoty

spojit pomocí operátorů spoj (JOIN). Vše obdobně aplikujte i u další početních operací. Zkontrolujte s níže uvedeným kompletním zdrojovým kódem.

Obrázek 53 Kalkulačka - zdrojový kód

3.8 Metodický list 8

Název

Skupinový závěrečný projekt

Stručná anotace

Studenti se rozdělí do skupin a budou mít za úkol vytvořit závěrečný projekt. Projekt musí být na principu počítačové hry, kdy hráč pomocí myši či klávesnice může daným způsobem ovlivňovat její dění.

Časová dotace

2x 45 minut

Vstupní znalosti

- základní znalost práce s počítačem
- pokročilá práce se všemi probranými algoritmickými prvky

Pomůcky

- počítač s internetovým připojením a aktualizovaným internetovým prohlížečem

Kognitivní cíle

- student se orientuje ve zdrojovém kódu Scratch
- student aktivně spolupracuje s ostatními v týmu
- aktivně se podílí na práci projektu s originálním obsahem

Klíčové slova

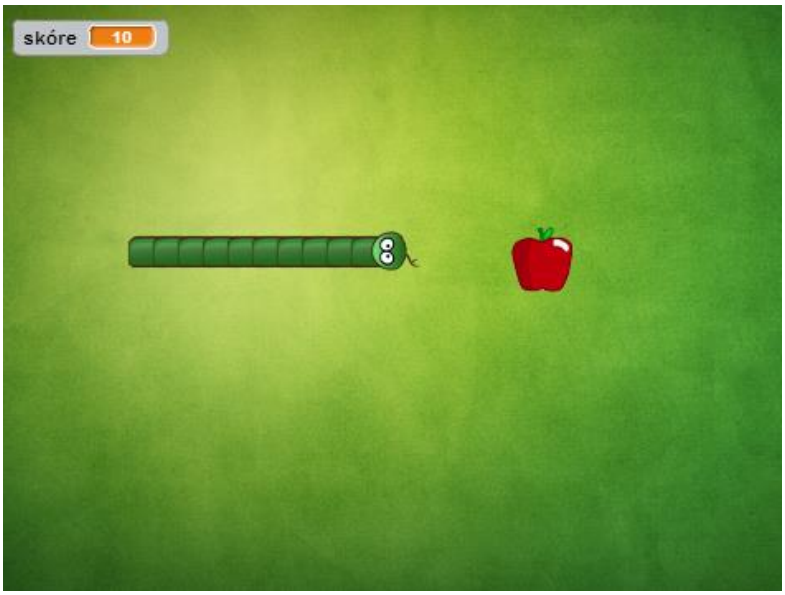
vývojové prostředí Scratch, algoritmus, počítačová hra, skupinová práce

Rozpracované téma

Cíl závěrečného metodického listu bude zaměřen na další rozšiřování získaných poznatků, a to především pomocí skupinového projektu. Studenti se tedy nejprve rozdělí do skupin, počet je závislý na počtu žáků ve třídě. Sdělte jim, že jejich úkolem bude vytvořit hru, která bude založena na motivech již existující hry, anebo mohou popřemýšlet a vytvořit zcela vlastní pravidla i grafické zpracování, což zohledněte v závěrečném hodnocení!

- Had - projekt inspirovaný kultovní hrou z doby první digiher a mobilních telefonů
- Libovolný výběr – vhodné spíše pro více zkušené studenty

Vzorový projekt Had je inspirovaný kultovní hrou Snake z doby prvních mobilních telefonů. Po neznalé sdělte jednoduchý princip, na kterém hra funguje. Po spuštění hry se začne had pohybovat a jeho pohyb již nelze zastavit. Had sbírá potravu a současně s tím roste i jeho délka. Během sběru nesmí narazit do zdi bludiště, v němž se pohybuje, pokud ano, tak v tom momentu hra končí.



Obrázek 54 Had - inspirativní vzhled herní scény

Algoritmus Hada si rozdělte do několika částí:

- tvorba ovládání
- pohyb hada (rychlost, změna délky)
- sběr potravy
- game over

Tvorba ovládání

Algoritmus pro ovládání hada pomocí klávesových šipek je složen z čtyř podmínek, přičemž každá z nich se zabývá pohybem, kterému odpovídá stisk specifické klávesy (šipka nahoru, dolů, vpravo, vlevo). Dále je nutno vytvořit hodnotu směr, která bude ve spojení s další částí programu nastavovat výsledný směr pohybu.

Obrázek 55 Had - tvorba ovládání

Pohyb hada

Pro vytvoření pohybu je nutno vytvořit další tři proměnné, a to v první řadě proměnnou, která bude pracovat s délkou hada. Hodnotu proměnné budete ovlivňovat až v další části programu. Druhá proměnná, která bude ovlivňovat rychlost a další bude mít na starosti proměnu skóre. Při spuštění je nastavte na výchozí hodnoty. Nezapomeňte, že hodnoty zásadně ovlivní obtížnost hry, kupříkladu zvýšením rychlosti hada. Doplňte příkaz, který způsobí to, že had se začne pohybovat vždy ze středu scény. Poté vložte nekonečný cyklus, který bude ovlivňovat jak změnu skóre, které se bude měnit při každém sběru potravy. Připomeňte studentům, že proměnná skóre je závislá na proměnné délka tzn. pokud had sebere potravu, zvýší hodnota skóre o hodnotu 1 a současně se o stejnou hodnotu také změní jeho délka. Poté doplňte vhodně klonování těla a otestujte funkčnost dosavadních částí programu. Pro inspiraci při tvorbě programu slouží níže uvedený zdrojový kód.

Obrázek 56 Had - pohyb

Sběr potravy

Nejprve si nahrajte nové sprita, který bude vypadat jako hadova potrava. Algoritmus toho sprita, bude nejdříve náhodně generovat souřadnice, kde se objeví. Poté bude obsahovat nekonečný cyklus, který bude obsahovat vnořený cyklus s podmínkou, která bude kontrolovat, zdali se hlava hada nedotkla potravy. Doplňte cyklus o příkaz, který při splnění podmínky, zahraje zvuk a následně se hodnota proměnné délka změní o hodnotu 1.

Obrázek 57 Had - sběr potravy

Game over

Vytvořte novou scénu, kterou opatřete různými vizuálními prvky. Poté nastavte, aby při spuštění hry se automaticky načetla herní scéna. Dále využijte scénáře události - konkrétně příkazů pro zasílání zpráv. Tímto způsobem ošetřete všechny části programu – kupříkladu po obdržení zprávy gameover se potrava skryje a všechny probíhající příkazy se zastaví. Zcela nutné je doplnit první část programu (pohyb hada) podmínkou, která provede příkaz **rozešli všem gameover a čekej**, pokud se had dotkne okraje scény. Obrázek 58 slouží pro představu, jak graficky zatraktivnit výslednou scénu.

Obrázek 58 Had – scéna game over

Další možnosti rozšíření spočívá v tom, že se kupříkladu had nesmí také dotknout svého těla, pokud se jej dotkne, hra ihned skončí. Dále lze přidat algoritmus, který při každém přičteném bodu zvýší rychlost hada o určitou hodnotu. V neposlední řadě lze přidat atraktivní žebříček hráčů s nejvyšším skórem.

4 REALIZACE A VYHODNOCENÍ

Metodické listy jsem realizovala během pedagogické praxe, vzhledem k nabytému tematickému plánu jsem měla bohužel k dispozici pouze dvě vyučovací hodiny. Realizace se účastnilo osmnáct studentů prvního ročníku oboru Hotelnictví. Součástí výuky byl obsah metodického listu č. 1, který slouží k seznámení s vývojovým prostředím Scratch a následoval metodický list č. 2 pomocí kterého se studentům představí problematika cyklů. Veškerá výuka probíhala zcela podle vypracovaných metodických listů a k ověření efektivity práce studentů jsem využila jak pozorování při samotné výuce, tak také vyhodnocení samostatných prací a v neposlední řadě, také jednoduchý dotazník, kterým jsem studentům rozdala po skončení výuky.

4.1 Realizace metodického listu č. 1

Počátkem vyučovací hodiny jsem se dotázala, zdali některý ze studentů zná pojem algoritmus a pokud ano, tak jestli dokáže tento pojem ostatním objasnit. Ze sedmnácti studentů se přihlásili dva studenti a trochu rozpačitě také jedna studentka. Veskrze jejich odpovědi obsahovaly správný popis, který po mé výzvě doplnili o vhodný příklad - algoritmus čištění zubů, algoritmus vázání kravaty a návod na pletení francouzského copu. Doplnila jsem uvedené příklady o příklad z metodického listu – algoritmus žehlení košile a zapsala jsem slovní zápis tohoto algoritmu přímo na tabuli. Následně jsem vyzvala studenty, kteří vyslovili své příklady algoritmů, aby své nápady zapsali pomocí slovního zápisu algoritmu dle existujícího vzoru na tabuli. Ostatním studentům jsem mezitím zadala za domácí úkol, aby sestavili také vlastní slovní zápis libovolného algoritmu. Poté jsem představila on – line vývojové prostředí Scratch, a to ukázkou již existujících projektů. Největší ohlas měla hra Space Tunnel, kde pomocí klávesových šipek se pilotuje vesmírná loď, která se musí udržet ve dlouhém a klikatém tunelu. Obtížnost se neustále zvyšuje, přičemž body jsou udělovány za ujetou vzdálenost (dostupné je i highscore). Hra končí v momentu, kdy se loď dostane mimo vyznačený tunel. Dále jsem požádala studenty, aby se na stránkách (http://scratch.mit.edu/) zaregistrovali vlastní profil a mohli tak pracovat na svým projektech i z domu. Plynule jsme přesunuli na první cvičný příklad a studenti ihned zcela intuitivně skládali příkazy do sebe a tím vytvořili první algoritmický prvek, a to posloupnost příkazů. Následující samostatná práce měla za cíl vytvořit algoritmus pro nakreslení čtverce a poté jej zapsat pomocí příkazů Scratche. Studenti se nejprve chvíli zamýšleli, ale poté bez větší obtíží všichni vytvořili funkční program pro zakreslení čtverce do scény.

4.2 Realizace metodického listu č. 2

Při realizaci druhé metodického listu jsem stavěla na tom, že se studenti již orientují ve vývojovém prostředí Scratch, přičemž někteří z nich mi sdělili, že strávili odpoledne po první výuce na serveru, kde si prohlíželi a testovali již existující projekty. Vyučovací hodina začala kontrolou domácího úkolu (slovní zápis algoritmu) a následně zadání dalšího domácího úkolu, ve kterém studenti měli za úkol vytvořit pomocí obdobného využití příkazů program, který bude kreslit na scénu různé geometrické obrazce. Dále vyučovací hodina pokračovala vysvětlení cyklu s daným počtem opakování a také cyklu s podmínkou – označovaným také jako cyklus REPEAT – UNTIL. Studenti rozdíl v použití různých typů cyklů bez větší obtíží pochopili a vzápětí až na malé výjimky aktivně pracovali na cvičných příkladech. Nabyté znalosti ověřovala samostatná práce, kterou studenti samostatně přišli na princip vnořování cyklů do sebe.

4.3 Vyhodnocení realizace metodických listů

Výuka spjatá s výukovým programovacím jazykem Scratch se studentům jevila jako velmi intuitivní a zábavná. V následném dotazníku velká část studentů uvedla, že se algoritmizaci a programování nikdy nevěnovala, pouze dva studenti uvedli, že se ve volném čase věnují tvorbě webových stránek a občas vytvoří jednoduché skripty pomocí programovacího jazyka PHP. Dále všichni studenti, ale uvedli, že se s výukovým programovacím jazykem Scratch setkali poprvé. Je proto potěšující, že každý z nich byl schopen během 45 minut naprogramovat jednoduchý příklad, tudíž se zcela naplnil kognitivní cíl výuky. V průběhu metodického listu č.2 jsem dospěla k názoru, že dva cvičné příklady, jsou pro některé studenty na jednu vyučovací hodinu příliš málo – zbývající čas jsem musela doplnit dalšími příklady, které jsem musela pohotově analogicky odvodit od předchozích příkladů, což mě vnuknulo myšlenku, že by bylo vhodné každý metodický list doplnit o příklady pro studenty, kteří rychleji pracují. Tato úprava by jistě přišla i vhod, jelikož i každá z tříd pracuje v určitém tempu a není snadné určit bez provedení v praxi, kolik příkladů má metodický list obsahovat. Po úpravě by metodický list obsahoval více příkladů, které procvičují podobné příkazy a učitelé by si mohli sami zvolit, které příklady využijí. Během výuky inspirované metodickým listem č. 2 bylo zapotřebí chodit mezi studenty a příležitostně někomu poradit. Naproti tomu, někteří z nich se aktivně zapojili do výuku s nejrůznějšími dotazy ohledně využití dalších příkazů a následně tvořili ve volných chvílích vlastní program tím, že obdobně využívali příkazy z cvičných příkladů. Studenti ocenili také prostor pro rozvoj vlastní kreativity, ke kterému jsem se vybídla prostřednictvím domácího úkolu, kterými následně zaslali pro kontrolu (k nahlédnutí v příloze této práce).

ZÁVĚR

Moderní přetechnizovaný svět nabízí vedle řady výhod a vymožeností, současně také značné negativa. Jedním z nich je fakt, že děti jsou již v předškolním věku zahlcovány velkým množstvím právě technických vymožeností – mobilními telefony, počítači, tablety, herními konzolemi a tím pádem tráví v jejich společnosti několik desítek hodin denně. Takto strávený čas ve velké většině případů nedává dětem nic přínosného, ačkoliv existuje velká řada výukových možností, jak kupříkladu čas spojený s počítačem efektivně využít právě pro rozvoj a další rozšiřování logických hranic daného dítěte. Motivací pro tvorbu této diplomové práce sloužil především již výše zmíněný fakt, který dle mého názoru lze využít právě ku prospěchu těchto dětí.

Diplomová práce zhodnotila kvalitu současné výuky algoritmizace a programování na středních školách, a to formou vyhodnocení dotazníku. Oslovením několika desítek učitelů ze školské praxe, jsem získala detailnější pohled na názory, které mě ještě více utvrdily v myšlence, že je nutné atraktivním a současně i efektivním způsobem seznámit všechny studenty středních škol, alespoň se základními algoritmickými principy. Dle odpovědí respondentů je zřejmé, že na středních školách není dostatečný prostor věnovaný právě výuce algoritmizace a programovaná, a pokud ano, tak pouze pro vybranou skupinu studentů.

Práce se dále zabývala dostupnými algoritmickými nástroji, jenž jsou speciálně vytvořeny, právě pro výuku algoritmizace a programování. Mezi tyto nástroje patří Imagine Logo ve spojení s želví grafikou, klasika výukových nástrojů Karel, programovatelná robotická stavebnice Lego Mindstorms, zástupce představující práci českých programátorů – Baltík a v neposlední řadě moderní Scratch. Poté byla zvolena hodnotící kritéria, a to přívětivost algoritmického nástroje, možnost jeho návaznosti na vyšší programovací jazyky, jazyková mutace, typ dostupné licence, s tím spojená pořizovací cena nutná pro následné využívání a také zdali je druh verze off – line nebo on – line, tedy dostupná jako cloudová aplikace (poskytovaná jako služba). Důležitým aspektem byla, ale především šíře nabízených algoritmických vlastností a obzvlášť jejich názornost pro pochopení. Následně dle těchto kritérií byl zvolen jeden z nich, a to výukový programovací jazyk Scratch – především pro své jednoduché, intuitivní ovládání a také pro svou atraktivní on – line verzi, která je absolutně zdarma. Scratch je primárně určen studentům od šesti do šestnácti let bez předchozích programovacích zkušeností, proto při realizaci ve výuce doporučuji zvolit první až druhý ročník středoškolského studia. Celkově jsem vypracovala osm metodických listů využívajících právě vývojové prostředí Scratch, jejichž složitost

má logicky zvyšující se tendenci, což studenty provede od tvorby základní posloupnosti příkazů až po tvorbu replik kultovních počítačových her. Metodické listy mají sloužit jako vodítko pro učitele, kteří by chtěli zařadit moderní formu výuky algoritmizace do tematického plánu svého předmětu, jenž spadá podle RVP do vzdělávací oblasti Informačních a komunikačních technologií.

Následovala realizace, kterou jsem provedla v rámci souvislé pedagogické praxe. Vzhledem k nízké oborové časové dotaci předmětu Informační a komunikační technologie, a tedy k nabytému tematickému plánu výuky jsem měla vyhrazeny pro vlastní libovolnou výuku pouze dvě vyučovací hodiny v jedné třídě. Studenti po absolvování výuky pomocí dvou metodických listů byli schopni využít při vytváření projektů základní algoritmické struktury, a to posloupnost příkazů či jednoduché cykly. Závěrem není možné s jistotou říci, zdali metodické listy přispěli k lepšímu pochopení problematiky algoritmů, což by bylo možné pouze v případě, že by proběhla realizace celého konceptu na sebe navazujících metodických listů a nejlépe ve více třídách, aby mohlo dojít ke srovnání a následnému vyhodnocení.

Pevně doufám, že mnou vytvořené metodické listy ovlivní a poslouží řadě učitelům, kteří sbírají odvahu, aby zařadily výuku algoritmizace a programování do své výuky. Věřím, že se úsilí, které jsem vynaložila při tvorbě této diplomové práce, kvalitativně odrazí při mé další tvorbě metodických listů v rámci svého budoucího učitelského povolání.

SEZNAM POUŽITÉ LITERATURY

1. Česká škola: ICT ve škole - Výuka. [online]. [cit. 2015-02-09]. Dostupné z: http://www.ceskaskola.cz/2006/10/jiri-vanicek-imagine-logo-aneb-moderni.html
2. THOMAS, David, Chad FOWLER a Andrew HUNT. Programming Ruby 1.9 & 2.0: the pragmatic programmers' guide. xvi, 863 pages. Pragmatic programmers. ISBN 1937785491.
3. WHEATLEY, Grayson H. Constructivist perspectives on science and mathematics learning. Science Education [online]. 1991, [cit. 2015-02-17].
4. BLAHO, Andrej a Ivan KALAŠ. Imagine Logo: učebnice programování pro děti. Vyd. 1. Brno: Computer Press, 2006. 48 s. Česká škola (Computer Press). ISBN 80-251-1015-X.
5. Introduction to programmming using FORTRAN 95 [online]. 2011. vyd. [cit. 2015-02-21]. Dostupné z: http://www.fortrantutorial.com/documents/IntroductionToFTN95.pdf
6. VIRIUS, Miroslav. Základy algoritmizace. Vyd. 2., přeprac. Praha: Česká technika - nakladatelství ČVUT, 2008, 265 s. ISBN 978-80-01-04003-4.
7. Richard E. Pattis. [online]. [cit. 2015-03-01]. Dostupné z: http://en.wikipedia.org/wiki/Richard_E._Pattis
8. ROBERTS, Eric. KAREL THE ROBOT LEARNS JAVA [online]. Stanford University, 2005 [cit. 2015-03-03]. Dostupné z: https://web.stanford.edu/class/cs106a/materials/karel-the-robot-learns-java.pdf
9. Nápověda k programu KAREL [online]. [cit. 2015-03-03]. Dostupné z: http://petr.lastovicka.sweb.cz/ostatni.html#karel
10. FERRARI, Mario, Giulio FERRARI, David ASTOLFO a Mario FERRARI. Building robots with Lego Mindstorms NXT. Updated ed. /. Oxford: Elsevier Science [distributor], c2007, 447 p. ISBN 1597491527.
11. Robotc [online]. [cit. 2015-03-10]. Dostupné z: http://en.wikipedia.org/wiki/Robotc
12. PELCOVÁ, Kateřina. Rozvoj algoritmického myšlení pomocí Lego Mindstorms. [online]. [cit. 2015-03-10]. Dostupné z: http://spomocnik.rvp.cz/clanek/17073/
13. Česká škola robotiky [online]. [cit. 2015-03-10]. Dostupné z: http://www.ceskaligarobotiky.cz/o-fll/v-cem-se-soutezi
14. PECINOVSKÝ, Rudolf. Baltík – Učebnice programování nejen pro děti. SGP Systems, 2001, 216 s.

15. ŠIMKOVÁ, Kristýna. Metodické listy pro výuku programování žáků 2. stupně základních škol. Olomouc, 2013. Bakalářská práce. Univerzita Palackého v Olomouci. Fakulta pedagogická. Vedoucí práce Mgr. Jan Kubrický.

16. Baltík [online]. [cit. 2015-03-14]. Dostupné z: http://cs.wikipedia.org/wiki/Balt%C3%ADk

17. SGP SYSTEMS. SGP Baltie 4 C# Screenshots [online]. [cit. 2015-03-14]. Dostupné z: https://www.sgpsys.com/en/Images/screenshots_b4/ConsoleCSharpMode.png

18. Scratch [online]. [cit. 2015-03-18]. Dostupné z: http://cs.wikipedia.org/wiki/Scratch

19. SVATAVA, Kašpárková. Učení a vyučování. Univerzita Tomáše Bati ve Zlíně, 2013. ISBN 978-80-7454-298-5.

20. BOTEK, Zdeněk. Základy informačních technologií. Zlín: Univerzita Tomáše Bati ve Zlíně, 2013.

21. VLADIMÍRA SEHNALOVÁ, Anna Závadská. Metodika výuky informatiky na 2. stupni základních škol a středních školách z pohledu pedagogické praxe - náměty pro začínajícího učitele. 1. vyd. Ostrava: Ostravská univerzita v Ostravě, 2010. ISBN 978-807-3688-912.

22. PECINOVSKÝ, Rudolf. Učebnice programování - základy algoritmizace. 1 vyd. Praha: Grada Publishing, 1997, 177 s. ISBN 80-716-9577-7.

23. Operator (computer programming) [online]. [cit. 2015-04-01]. Dostupné z: http://en.wikipedia.org/wiki/Operator_(computer_programming)

24. Portable Network Graphics [online]. [cit. 2015-04-09]. Dostupné z: http://cs.wikipedia.org/wiki/Portable_Network_Graphics

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

PNG - Portable Network Graphics

SaaS - Software as a Service

RVP - Rámcový vzdělávací program

SEZNAM OBRÁZKŮ

Obrázek 1 Imagine Logo - Řada čísel 15
Obrázek 2 Imagine Logo - Příkaz cyklu 16
Obrázek 3 Imagine Logo - Rekurzivnost procedur 16
Obrázek 4 Karel – Příkaz čtverec 17
Obrázek 5 Karel - Jednoduchá rekurze 18
Obrázek 6 Lego Mindstorms - Bloky akcí 19
Obrázek 7 Baltík - Program KOLONA AUT 20
Obrázek 8 Baltík – Řešení KOLONA AUT 20
Obrázek 9 Prostředí SGP Baltie 4 C# PRO (16) 21
Obrázek 10 Scratch - nakreslení čtverce 22
Obrázek 11 Popis prostředí Scratch 2.0 29
Obrázek 12 První program 29
Obrázek 13 Samostatná práce 1 30
Obrázek 14 Řešení samostatné práce 1 30
Obrázek 15 Trojúhelník 32
Obrázek 16 Cyklus 32
Obrázek 17 Cyklus obrazec 32
Obrázek 18 Syntaxe cyklu REPEAT – UNTIL (19) 33
Obrázek 19 Cyklus REPEAT – UNTIL 33
Obrázek 20 Samostatná práce - vnořený cyklus 34
Obrázek 21 Nekonečný cyklus 35
Obrázek 22 Nekonečný cyklus 2 35
Obrázek 23 Práce s proměnnou 36
Obrázek 24 Počítadlo 37
Obrázek 25 Počítadlo – domácí úkol 37
Obrázek 26 Samostatná práce – proměnná 38
Obrázek 27 Kalendář 38
Obrázek 28 Zdrojový kód - kalendář 39
Obrázek 29 Základní operátory 40
Obrázek 30 Blok spoj (join) 41
Obrázek 31 Logické operátory 41
Obrázek 32 Generování čísel 42

Obrázek 33 Samostatná práce - násobilka 42
Obrázek 34 Moorhunh ve Scratch 44
Obrázek 35 ScratchMoorhunh - první část 45
Obrázek 36 ScratchMoorhunh - druhá část 45
Obrázek 37 ScratchMoorhunh – algorimus střelby 45
Obrázek 38 ScratchMoorhunh - klonování 46
Obrázek 39 ScratchMoorhunh 47
Obrázek 40 ScratchMoorhunh - rozšíření výsledné hry 47
Obrázek 41 Podmínka IF-ELSE 49
Obrázek 42 IF-ELSE a logický operátor AND 49
Obrázek 43 IF-ELSE a logický operátor OR 50
Obrázek 44 IF-ELSE a logický operátor NOT 50
Obrázek 45 Úplný podmíněný příkaz - syntaxe 50
Obrázek 46 IF-THEN-ELSE 51
Obrázek 47 Lichost a sudost čísel 51
Obrázek 48 Lichost, sudost čísel a operátor JOIN 51
Obrázek 49 Nastavení středu tužky 52
Obrázek 50 Tloušťka tužky s využití proměnné 53
Obrázek 51 Reakce na obdrženou zprávu 53
Obrázek 52 Ukázka kalkulačky 55
Obrázek 53 Kalkulačka - zdrojový kód 56
Obrázek 54 Had - inspirativní vzhled herní scény 58
Obrázek 55 Had - tvorba ovládání 59
Obrázek 56 Had - pohyb 60
Obrázek 57 Had - sběr potravy 60
Obrázek 58 Had – scéna game over 61

SEZNAM GRAFŮ

Graf 1 – Typ střední školy 11
Graf 2 - Počáteční výuka algoritmizace 11
Graf 3 - Výukové programovací jazyky 12
Graf 4 - Algoritmizace pro všechny studenty 13

SEZNAM PŘÍLOH

Příloha 1 – Dotazník

Výuka algoritmizace na středních školách

Vyučuji na

- ☒ gymnáziu
- ☒ střední odborné škole s technickým zaměřením
- ☒ střední odborné škole s jiným zaměřením (zdravotnictví, hotelnictví apod.)
- ☒ středním odborném učilišti
- ☒ Jiné:

Pro počáteční výuku algoritmizace je lepší využít

- ☒ výukový programovací jazyk
- ☒ objektový programovací jazyk
- ☒ strukturovaný programovací jazyk

Který programovací jazyk ve výuce používáte převážně?

☒

A z jakého důvodu?

☒

Z výukových programovacích jazyků mě nejvíce oslovil

- ☒ Baltík
- ☒ Karel
- ☒ Imagine Logo
- ☒ Scratch
- ☒ Kodu Game Lab

- ☒ Jiné:

Jaké jsou výhody výukových programovacích jazyků?

- ☒ vizuální odezva
- ☒ pochopení principů algoritmů bez znalosti složité syntaxe
- ☒ Jiné:

Jaký je Váš názor na využití programovatelné robotické stavebnice Lego Mindstorms ve výuce?

- ☒ Aktivně ji využíváme
- ☒ Nemám s ní žádné zkušenosti
- ☒ Jiné:

Dle mého názoru, většinu začátečníků od programování odradí ....

- ☒ nutnost dodržovat přísnou syntaxi
- ☒ složitost pochopení struktury kódu
- ☒ komplexnost vývojového prostředí
- ☒ Jiné:

Je základní výuka algoritmizace důležitá pro všechny studenty?

- ☒ Ano
- ☒ Ne

A z jakého důvodu ...

Příloha 2 – Dotazník pro studenty

EVALUAČNÍ DOTAZNÍK
pro studenty

1. Jak celkově hodnotíte výuku?

1 2 3 4 5

2. Splnila výuka Vaše očekávání?

1 2 3 4 5

3. Získal(a) jsi nové znalosti a dovednosti?

ano možná ano částečně spíše ne ne

4. Byli jste spokojeni s učitelem a jeho přístupem k výuce?

ano možná ano částečně spíše ne ne

5. Co si před touto výukou věděl(a) o algoritmizaci?

---

6. Setkali jste se již s dříve s prog. jazykem Scratch?

ano ne

SCRATCH
imagine • program • share

Příloha 3 – Domácí úkoly