Vybrané příklady k výuce nové informatiky na základních školách

Bc. Zdeněk Vrbík

Diplomová práce
2024

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2023/2024

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Zdeněk Vrbík**
Osobní číslo:	**A22338**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Učitelství informatiky pro střední školy**
Forma studia:	**Prezenční**
Téma práce:	**Vybrané příklady k výuce nové informatiky na základních školách**
Téma práce anglicky:	**Selected Exercises for Teaching new Informatics in Primary Schools**

Zásady pro vypracování

1. Prostudujte dostupné sbírky příkladů pro výuku programování na základních a středních školách.
2. Vyberte obtížnější úlohy, které v zadání nemají dostatečně popsán způsob řešení a vypracujte k nim metodické pokyny pro učitele.
3. Vypracované úlohy seřaďte do tematických celků, ke každému celku vypracujte úvodní vysvětlující text.
4. Otestujte vaši sbírku příkladů v praktické výuce informatiky na ZŠ, nebo v kroužku programování.
5. Zdokumentujte a do své práce zapracujte zpětnou vazbu z praktické výuky.

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

Seznam doporučené literatury:

1. BELL, Tim; WITTEN, Ian H.; FELLOWS, Mike. CS Unplugged. Christchurch, 2015. vydáno vlastním nákladem.
2. BLAHO, Andrej; KALAŠ, Ivan. Imagine Logo – Učebnice programování pro děti. Computer Press, 2006. ISBN 80-251--10-15-X.
3. HYLMAR, Radek. Programování pro úplné začátečníky. Computer Press, 2009. ISBN 978-80-251-2129-0.
4. PELÁNEK, Radek. Programátorská cvičebnice. Computer Press, 2012. ISBN 978-80-251-3751-2.
5. BERTRAND, Yves. Soudobé teorie vzdělávání. Portál, 1998. ISBN 80-7178-216-5.

Vedoucí diplomové práce: **Ing. Tomáš Dulík, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **5. listopadu 2023**
Termín odevzdání diplomové práce: **13. května 2024**

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
DĚKAN
L.S.

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 5. ledna 2024

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové 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 práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji diplomovou 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 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 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 práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové 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 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, dne …….

Zdeněk Vrbík, v. r.
podpis studenta

ABSTRAKT

Tato diplomová práce v úvodu teoretické části analyzuje aktuální situaci stran výuky informatiky v českém základním školství a to mj. i po stránce materiálně didaktických prostředků, technického vybavení, finanční situace a vyhlídek na nejbližší období. V praktické části pak přináší vybrané příklady k výuce tematických celků „Data, informace a modelování“ a „Algoritmizace a programování“ podle RVP pro základní vzdělávání. Uvedené příklady lze prakticky realizovat buď zcela bez jakéhokoliv technického vybavení, nebo jen se základním vybavením pro výuku ICT, které je na školách běžně k dispozici.

Klíčová slova:

výuka informatiky, učební pomůcky, informatické myšlení, algoritmizace, kódování, modelování, želví grafika, dvojková soustava, komprese dat, třídění, řešení problémů

ABSTRACT

In the introduction to the theoretical part, this diploma thesis analyzes the current situation of computer science teaching in Czech elementary schools, including in terms of didactic resources, technical equipment, financial situation and prospects for the near future. In the practical part, it presents selected exercises for teaching the thematic units “Data, information and modeling” and “Algorithmization and programming” according to Framework Educational Program for basic education. The given exercises can be practically implemented either completely without any technical equipment, or with only basic equipment for teaching ICT, which is normally available in schools.

Keywords:

teaching informatics, teaching aids, computational thinking, algorithmization, coding, modeling, turtle graphics, binary numeral system, data compression, sorting, problem solving

Děkuji vědoucímu diplomové práce Ing. Tomáši Dulíkovi, Ph.D. za jeho cenné rady a připomínky a vstřícný přístup.

MOTTO:

„*Nedostatky ve školství nemohou být vyřešeny technologiemi. Jakékoli množství počítačů situaci nezlepší… Můžeme uložit všechny vědomosti na CD-ROMy. Můžeme dát WWW server na každou školu – nic z toho není špatné. Špatné je to až v tom okamžiku, když si začneme myslet, že jsme udělali něco pro vyřešení problémů se vzděláváním*.“

Steve Jobs

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

OBSAH

ÚVOD

Tato práce přináší ve své teoretické části mj. výklad základních teorií k algoritmům a algoritmizaci, kódování a modelování, i několik historických zajímavostí a v praktické části mj. několik konkrétních tipů (k výkladu i ověření porozumění) pro výuku základů informatiky na základní škole.

Dostupné učební materiály pro výuku programování na základních školách (bod zadání 1) – jsou popsány v podkapitole 2.6. Vybrané obtížnější úlohy, které nemají v zadání dostatečně popsán způsob řešení s vypracování úvodního vysvětlujícího celku (body zadání 2 a 3) jsou popsány v závěrečné, 10. kapitole. Poznatky získané z praktického testování při výuce na ZŠ (body zadání 4 a 5) jsou popsány v praktických poznámkách a závěrečném shrnutí na konci každé kapitoly v praktické části.

V práci jsou, vedle didaktické teorie, zahrnuta i témata týkající se, kromě tematického celku „Algoritmizace a programování“, i druhého Rámcovým vzdělávacím programem (RVP) v rámci „nové“ informatiky nově zavedeného tematického celku „Data, informace a modelování“. Úvod praktické části je věnován výuce informatiky na 1. stupni, tj. ve 4. a 5. ročníku, a je výrazně inspirován novozélandskou koncepcí výuky základů informatiky bez počítače.

I. TEORETICKÁ ČÁST

1 INFORMATIKA V ČESKÉM ŠKOLSTVÍ

„*Technologický rozvoj dvacátého* [a jednadvacátého] *století poznamenal stejně jako ostatní sociální instituce i školy. Tento vliv byl zřetelný ve dvou rovinách - jednak v rovině zavádění konkrétních technologických prostředků, jednak v rovině různých utopických projektů, které se zrodily z nadšení nad velkým potenciálem změn*.“ [1, s. 89] „*…model, který v jisté chvíli dává určitou představu o stavu poznání v dané disciplíně, je o několik měsíců později překonán. Je tím nastolen důležitý problém aktuálnosti vzdělávání. Důsledkem je, že zacházení s poznatky a dovednostmi a jejich transfer vyžadují, aby byl vzat v úvahu jejich dynamický a měnící se charakter a aby bylo vytvořeno prostředí, které se bude rovněž měnit a vyvíjet*.“ [1, s. 108]

Takové prostředí vytvořila v českém školství tzv. *kurikulární reforma*, jejíž (postupný) start do reálné praxe se datuje do let 2004/2005. Kurikulární reforma zavádí prostřednictvím Školského zákona (561/2004 Sb.) Rámcové vzdělávací programy (RVP), které dávají školám jistou autonomii v aktualizaci svých Školních vzdělávacích programů (ŠVP) podle jejich zaměření, příp. technických a jiných možností. „*Škola si* [tak] *nemůže dovolit zaujmout pozici mrtvého brouka nebo si s pedagogickou reformou jen pohrávat, aby vytvořila iluzi změny*.“ [1, s. 223]

„*Současná společnost … stále více závisí na technice a přehled v oboru informatiky je proto užitečný pro každého*.“ [2, s. 14] Podle nového RVP vydaného v lednu 2021 MŠMT ČR [3] musely základní školy počínaje školním rokem 2023-2024 nově zavést výuku informatiky na 1. stupni a na 2. stupni počínaje školním rokem 2024-2025 musí všechny základní školy přejít na tzv. „novou“ informatiku, která neobnáší uživatelské znalosti ICT, jak tomu bylo doposud, ale výuku základů algoritmizace a programování, kódování, šifrování, modelování apod. Tato diplomová práce přináší několik konkrétních řešených příkladů, jejichž zapojení do výuky není nijak náročné na speciální technické vybavení.

Rozhodnutí MŠMT ČR aktualizovat kurikulární obsah RVP pro výuku informatiky je motivováno snahou o to, aby vzdělávání reagovalo na potřeby praxe a aby škola rozvíjela kompetence žáků v oblastech, které jsou, a v dlouhodobé perspektivě lze očekávat, že i nadále budou, žádány na trhu práce. Toto rozhodnutí časově koresponduje s nástupem tzv. Průmyslu 4.0, tj. čtvrtou průmyslovou revolucí, která se vyznačuje intenzivním „*…využitím*

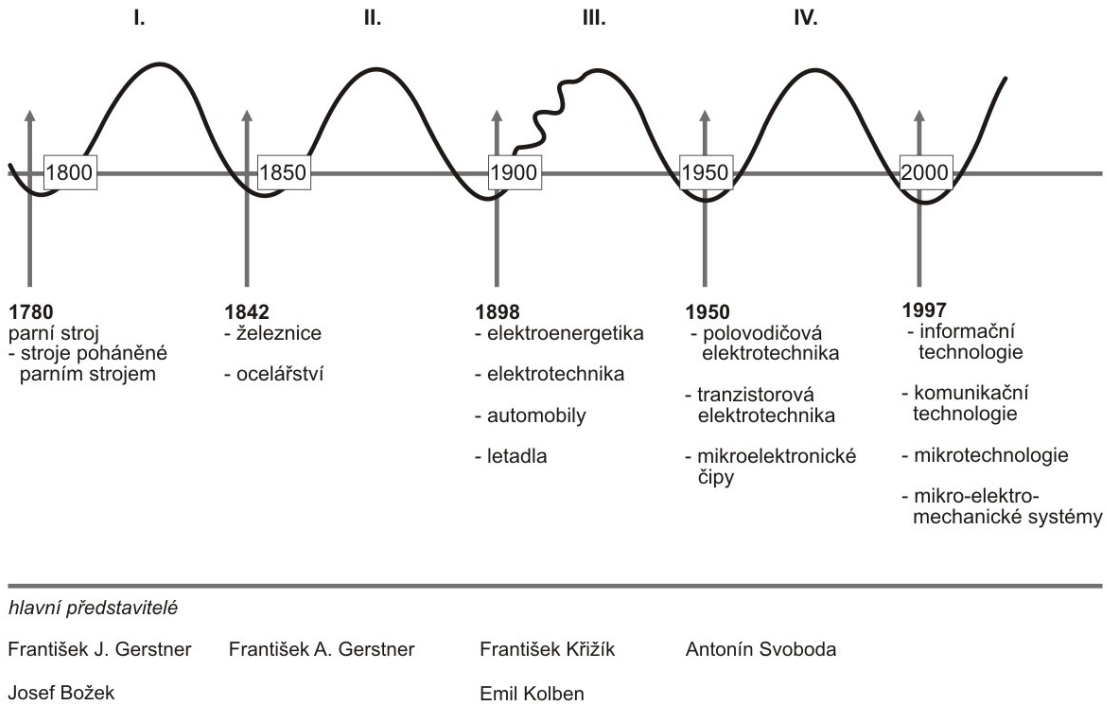
informačních a komunikačních technologií v pracovním prostředí, [kdy] *stroje jsou doplněny o síťové připojení na internet*.“[1] [4]

Toto rozhodnutí o aktualizaci obsahové náplně RVP pro výuku informatiky zároveň časově zapadá, jak ukazuje Obrázek 1, do tzv. V. Kondratěvovy inovační vlny. „*Ruský a později sovětský nemarxistický*[2] *ekonom …, teoretik plánování a statistik Nikolaj Dmitrijevič Kondratěv (1892-1938) na základě statistické analýzy formuloval a rozpracoval hypotézu dlouhých vln v ekonomickém vývoji. Popsal tak cyklus, který … s udivující přesností odráží vývoj několika minulých století*.“ [5, s. 55] Kondratěvova vlna je komplex inovací určitého charakteru, které probíhají po dobu zhruba půlstoletí.

Jisté zpoždění reakce nejen vzdělávacího systému na aktuální potřeby praxe vysvětluje „… *americký filozof, fyzik a teoretik vědy a vědeckého poznání Thomas Samuel Kuhn (1922-1996) ve své světově uznávané knize Struktura vědeckých revolucí* (The Structure of Scientific Revolutions, 1962), [v níž] *ukazuje, že každý významný průlom v oblasti vědeckého snažení je nejprve porušením tradice starého způsobu myšlení. Než je nová skutečnost vědeckou obcí rozpoznána nebo dokonce akceptována, trvá to 30 až 50 let. V podstatě tak dlouho, dokud nevyroste a neprosadí se nová vědecká generace, která je schopna absorbovat nové poznatky a prosadit nové myšlení*.“ [6, s. 18-19]

Zpoždění reakce vzdělávacího systému na aktuální potřeby praxe není jediným problémem současného českého školství (zejména základního a středoškolského vzdělávání), dalším, úzce souvisejícím s předchozím, je problém nedostatku kvalifikovaných učitelů informatiky. [7] Těžiště tvorby učebních materiálů (nejen pro výuku informatiky) se ale stále více přesouvá do online prostředí s interaktivními výukovými programy, což poněkud oslabuje roli učitele ve výuce. [1, s. 115-116] Ačkoliv jde o moderní trend ve školství, najdou se i kritikové tohoto aktuálního trendu, kteří zastávají názor, že „ …*role učitele jako ‚technika‘, jehož hlavním úkolem je zavádět do výuky materiály ‚někoho jiného’, se ukázala jako neefektivní a bez potenciálu k dlouhodobější změně či vývoji*. [8, s. 21]

[1] V souvislosti se síťovým propojením různých fyzických zařízení se hovoří o tzv. „internetu věcí“, což je tzv. kalk, tj. doslovný překlad anglického *Internet of Things* (IoT).
[2] Dostal se do konfliktu se sovětskou mocí, byl dvakrát souzen. V roce 1930 byl zatčen a deportován do GULAGu, kde také zemřel.



Obrázek 1. Kondratěvovy vlny [9, s. 46]

Obrázek 1 uvádí jako hlavního představitele IV. Kondratěvovy inovační vlny u nás profesora Antonína Svobodu (1907-1980), českého vynálezce a počítačového vědce, konstruktéra prvních československých samočinných počítačů SAPO a EPOS 1, profesora na University of California v Los Angeles, autora jedné z prvních knih o počítačích, *Computing mechanisms and linkages*, která v roce 1946[3] vyšla v USA, v Anglii, v Číně. [10, s. 169]

Seznam žáků a spolupracovníků profesora Antonína Svobody je úctyhodně dlouhý a zahrnuje několik uznávaných vědců světového formátu, mezi nimi např. John von Neumann (1903-1957), Norbert Wiener (1894-1964), Claude E. Shannon (1916-2001) a další. [10, s. 169] „*Dnešní mladí počítačoví nadšenci nemají často ani tušení, že Československo bylo kdysi v počítačové technice na špičce vývoje. Obdivují počítače přicházející zpoza Atlantického oceánu a nevědí, že na konstrukci mnoha z nich se dodnes podílejí vědci a inženýři, kteří se to naučili v Praze*.“ [10, s. 171]

[3] Norbert Wiener publikuje svoji učebnici kybernetiky, *Cybernetics: Or Control and Communication in the Animal and the Machine*, v níž ve 2. pol. 20. stol. hledala inspiraci i teoretická pedagogika, „až“ v roce 1948. [1, s. 100]

V době, kdy Československo drželo v oblasti vývoje výpočetní techniky krok se světem, měl profesor Svoboda ideu: „*Tak jako je Švýcarsko velmocí v hodinkách, mohlo by být Československo velmocí v počítačích*.“ [10, s. 169] Podobný cíl si po 70 letech zřejmě klade i revize RVP v oblasti výuky informatiky.

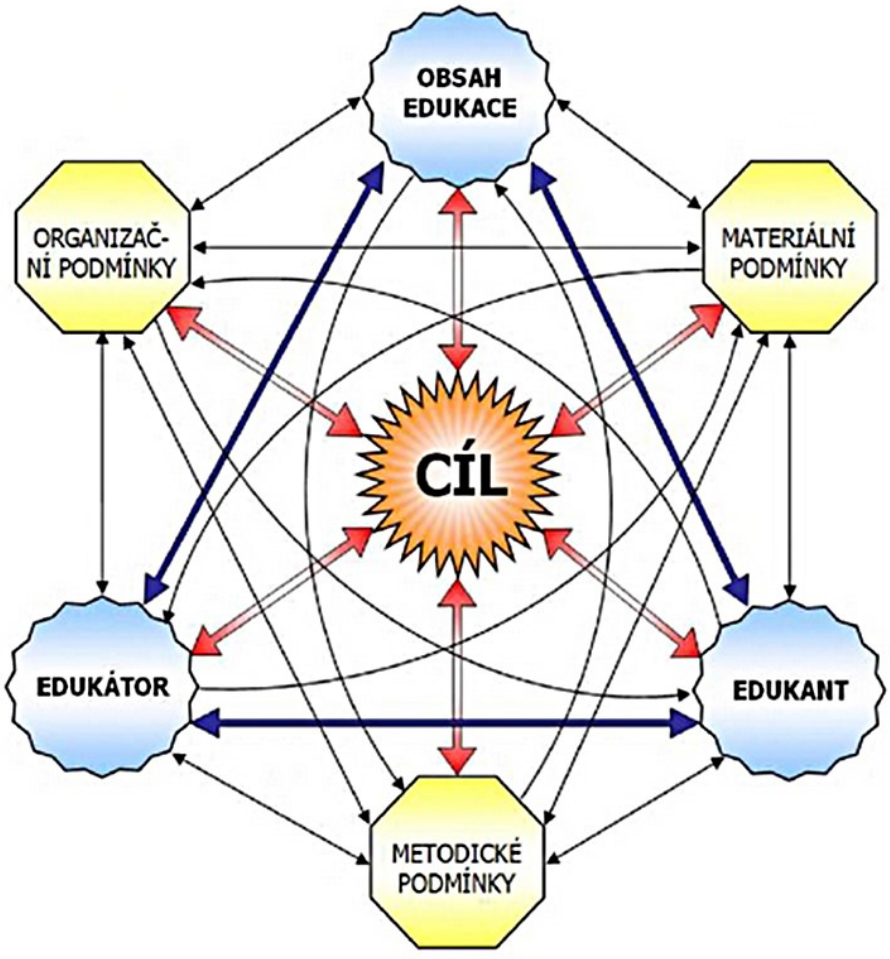
V praktické části této práce je několik tipů pro výuku základů informatiky na základní škole, ale jistě bude přínosná i pro učitele na středních školách, protože podobně jako ve vývojově-psychologické rovině „*…k plnému porozumění starším dětem potřebuje učitel na druhém stupni vědět něco o formativních vlivech, které působily na děti během raného období jejich života, zatímco učitel na prvním stupni potřebuje vědět něco o problémech, které na děti, o něž pečuje, teprve čekají, aby jim co nejvíce pomohl vytvořit dovednosti a strategie nezbytné k jejich zvládnutí*,“ [11, s. 16] by učitel na střední škole měl mít přehled o tom, co a jak se učí na základní škole, aby věděl, co žáci, kteří k němu přicházejí do prvních ročníků, znají a umí, a jakými metodami se to učili.

V této souvislosti lze hovořit o jakémsi širším pojetí pojmu prekoncept, který odborná pedagogická literatura definuje jako „*…vstupní představy, vědění žáků vzniklé na základě jejich zkušeností s danými jevy v realitě, z televize, z činností s těmito jevy. Většinou jde o neúplné poznání, poznání jevové stránky, praktické funkce. Při práci s obsahem ve škole tyto prekoncepty vstupují do konfrontace s vědeckým vysvětlením daných jevů*.“ [12, s. 106]

2 POMŮCKY VE VÝUCE

„*Učební pomůcky jsou dnes neodmyslitelnou součástí edukačního procesu … To, že máme učební pomůcky k dispozici, ještě neznamená, že budou pro osvojování vědomostí, dovedností a postojů přínosem. Naopak jejich nevhodné použití může působit kontraproduktivně*.“ [13, s. 5]

Český pedagog Josef Hendrich (1888 – 1950) „*… tradiční didaktický trojúhelník*[4]*: obsah, učitel, žák doplnil o kategorie cíl, organizační podmínky, materiální podmínky a vyučovací metody a postupy. S ohledem … na poznatky získané z … literatury je možné uvést grafické znázornění edukačního procesu*,“ [13, s. 8-9] viz Obrázek 2.



Obrázek 2. Grafické znázornění edukačního procesu [13, s. 9]

[4] Tento trojúhelník je znám již z teorií německého pedagogika Johanna Friedricha Herbarta (1776 - 1841).

Materiální podmínky, viz Obrázek 2 vpravo nahoře, jsou dány materiálními prostředky, které dále dělíme do skupin:

- vyučovací pomůcky
- žákovské pomůcky (potřeby)
- učebny a jejich vybavení
- didaktická technika

Některá odborná literatura dělí pomůcky a potřeby na žákovské a pomůcky a potřeby pro pedagoga a mezi materiální prostředky zahrnuje i samotného pedagoga, viz např. [14, s. 5] „*Zdánlivě by bylo při obecných úvahách možné didaktickou techniku zařadit do kategorie zařízení výukových prostor*[5] *... S ohledem na její význam, specifické možnosti a univerzální použití ji většina autorů uvádí jako samostatnou skupinu materiálních didaktických prostředků*.“ [14, s. 6] Někteří autoři ale pracují s kategorií, kterou označují jako *technické výukové prostředky* a v jejím rámci rozlišují dvě podkategorie, a sice *didaktická technika* a *učební pomůcky*.

2.1 Didaktická technika a učební pomůcky

Slovní spojení didaktická technika „… *zahrnuje přístroje a technické systémy, které využíváme pro výukové účely. Pedagogovi umožňují prezentaci některých druhů učebních pomůcek, případně pomáhají zvyšovat účinnost výuky. V praxi jde o projekční, auditivní, nebo audiovizuální přístroje, které na základě splnění určitých předpokladů mohou umožňovat případně i multimediální výuku*.“ [14, s. 6]

Slovní spojení učební pomůcky představuje „… *učební informace, v podstatě vždy vybrané sofistikované obsahy výuky ... Ve své podstatě jsou nosiči předávaných obsahů. Některé pomůcky lze studentům prezentovat přímo – např. modely, učebnice, skripta. Některé učební pomůcky vyžadují vzhledem ke své povaze k prezentaci určitý prostředek – didaktickou techniku. Příkladem může být učební záznam jako učební pomůcka, která bude prezentována didaktickou technikou, např. MP3 přehrávačem*.“ [14, s. 6]

---

[5] tj. učebny a jejich vybavení
Důraz na výukové prostředí, jakožto na důležitý faktor výrazně ovlivňující výukový proces, kladl americký psycholog Burrhus Frederic Skinner (1904 - 1990), jeden z představitelů tzv. *technologické teorie* vzdělávání [1, s. 101], viz níže kapitola 2.3

Pro učební pomůcku je charakteristické:

- přibližuje to, co je daleké
- zvětšuje to, co je nepatrné
- zmenšují to, co je příliš veliké
- zpomalují to, co je příliš rychlé
- zrychlují to, co je pomalé
- odhalují to, co je skryté
- konkretizují to, co je abstraktní
- zpřítomňují to, co je minulé
- fixují to, co je prchavé
- zpřehledňují to, co je složité

[14, s. 7]

2.2 Počítač - učební pomůcka i didaktický prostředek

Jiná literatura zařazuje počítač do obou kategorií podle toho, probíhá-li výuka o počítači nebo s počítači,[6] ačkoliv připouští, že se obě kategorie „*…do určité míry prolínají a nejdou od sebe úplně oddělit*.“ [15, s. 7-8]

Jako učební pomůcka je počítač chápán „*…při výuce programování, obsluhy počítače, poznávání jednotlivých typů počítačů atd. Tato funkce přispívá ke zvýšení názornosti pomoci modelování, nejrůznějších simulací, grafiky a animací, dále napomáhá k zpřístupnění informací pomocí databanky a prezentace učební látky*.“ [15, s. 9]

„*Jako didaktický prostředek* [je počítač] *využíván při výuce s didaktickými programy a při spojení s diaprojektorem a třeba s i interaktivní tabulí …* [při klasické výuce, realizované] *v běžné třídě, jelikož v dnešní a ani nejbližší době nelze ve větším počtu předpokládat přesun výuky většiny předmětů různých zaměření (kromě informatiky) do tříd vybavených počítači, kde by všem žákům byla s počítači umožněna samostatná činnost. Ještě než se počítače začaly ve výuce využívat jako didaktická technika, bylo možné se setkávat s řadou pomůcek, jejichž činnost dnes dokáže počítač plně zastoupit*.“ [15, s. 9]

[6] častěji se lze setkat s formulací *výuka podporovaná počítačem*

2.3 Technologické teorie

Mezi nejvýznamnějšími soudobými proudy pedagogického myšlení, jak je definoval Yves Bertrand (*1945) ve své knize *Soudobé teorie vzdělávání* (Théories contemporaines de l´éducation, 1987), má k výuce informatiky zřejmě nejblíže tzv. *technologická teorie*. „*Technologické teorie, nazývané také technicko-systémové nebo systémové, obecně vzato zdůrazňují zdokonalení předávání informací použitím vhodných technologií. Slovu ‚technologie' zde musíme dát velmi široký význam. Zahrnuje postupy, se kterými se setkáváme v systémových přístupech a v koncipování výuky, ale i didaktické pomůcky pro komunikaci a pro zpracování informací: počítač*, …" [1, s. 17-18] Obnáší tedy spíše výuku s počítačem než výuku o počítači, viz předchozí podkapitola.

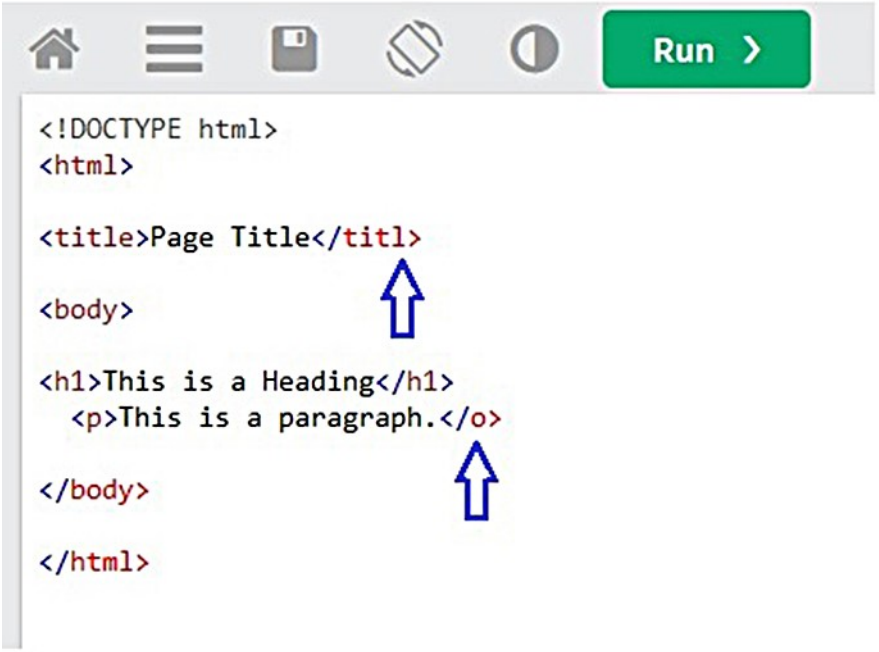
Psaní, byť jednoduchého, zdrojového kódu[7] ovšem v několika bodech odpovídá výčtu charakteristických prvků pro výuku organizovanou v duchu technologické teorie ve vzdělávání:

- žák formuluje hypotézy a počítač vytváří simulace, které pomáhají hypotézy ověřit nebo vyvrátit
- počítač zadá problém a pomáhá ho vyřešit
- počítač analyzuje řešení …
- počítač vykonává diagnostickou funkci - odhaluje a analyzuje žákovy omyly a chyby

[1, s. 101]

Příkladem, kdy počítač odhaluje chyby a upozorňuje na ně, může být třeba případ, kdy vývojové prostředí (Integrated Development Invironment - IDE) barevným zvýrazněním textu upozorňuje mj. na překlepy nebo chybějící znaky apod. Hovoří se o tzv. *zvýrazňování syntaxe* (syntax highlighting) „*Nejčastějšími chybami jsou totiž právě překlepy při realizaci programu* …" [16, s. 25] Některá vývojová prostředí mají tzv. *našeptávač* (autocomplete), který rovněž částečně eliminuje výskyt překlepů při psaní zdrojových kódů.

[7] Pecinovský definuje zdrojový kód jako „*text zadávaných příkazů*" [17, s 79]



Obrázek 3. Upozornění na chybějící znaky a překlepy barevným zvýrazněním v jednoduchém výukovém online vývojovém prostředí w3schools.com

„*Myšlenkovým zdrojem pro technologické teorie je* [psychologický směr] *behaviorismus. Behavioristické techniky se často používají tam, kde je zapotřebí dosáhnout změny specifického chování, zvláště se osvědčuje u dětí*.“ [18, s. 183] Behavioristické teorie, zcela logicky, ztotožnily počítače s vyučovacími stroji. [1, s. 101]

„*Psychologové* [zastávající behavioristický přístup k vyučování] *... vypracovali tzv. algoritmické (propracované) vyučování. L. N. Landa (1974) definuje algoritmus jako přesný a globální předpis, který musí vést k cíli definovanou řadu jednoduchých operací. Algoritmus tedy popisuje nejefektivnější řadu operací schopnou vyřešit matematický, chemický, fyzikální, lingvistický nebo jiný problém*.“[8] [1, s. 102]

2.4 Montessori

Používání (speciálních) pomůcek je jedním z typických rysů pedagogické metody Montessori. Italská lékařka a pedagožka Maria Montessori (1870-1952), autorka po ní

[8] Tato definice algoritmu Lva Nakhmanoviche Landy (1927 - 1999) z knihy *Algorithmization in Learning and Instruction* (Algoritmizace v učení a vyučování) je naprosto výstižná i pro potřeby informatiky.

pojmenované alternativní výchovně-vzdělávací koncepce, ovšem upozorňuje, že učitel by měl pomůcky vnímat jako něco, co usnadňuje žákovi učení a pochopení, nikoliv něco, co usnadňuje učiteli práci. [19, s. 153]

Pedagogika Montessori má i několik absolventů úspěšných v oblasti IT, „… *zakladatelé Amazonu, Jeff Bezos, a Googlu, Larry Page a Sergey Brin, jsou všichni tři bývalí montessoriovci a připisují velkou část svého úspěchu právě tomu, že chodili do montessoriovských škol. Říkají, že je tato výuková metoda naučila samostatnému myšlení a poskytla jim svobodu naplňovat svůj osobní příběh*.“[9] [20, s. 215]

Pro výuku informatiky používají školy mj. stavebnice LEGO Spike apod., které se používají i v běžných školách. Dochází tak, přinejmenším v oblasti výuky informatiky, ke sbližování klasických pedagogických metod, podle nichž se vyučuje na běžných státních školách, a škol Montessori. Na jedné straně je tomu tak proto, že školy Montessori se se svým kurikulem musí přizpůsobit požadavkům MŠMT ČR, a na druhé straně proto, že klasické školství se inspiruje metodou Montessori.[10]

2.5 Podpora rozvoje informatického myšlení

Ačkoliv se všechny aktuální pedagogické proudy shodují na tom, že pomůcky jsou důležitou součástí výuky, historie ale ukazuje, že i v materiálně skromných podmínkách lze vychovat člověka se základními kompetencemi pro život v moderním světě. Podobný názor vyjádřil Steve Jobs (1955-2011) v rozhovoru pro únorové číslo časopisu *Wired* z roku 1996, v němž mj. říká: „*Nedostatky ve školství nemohou být vyřešeny technologiemi. Jakékoli množství počítačů situaci nezlepší… Můžeme uložit všechny vědomosti na CD-ROMy. Můžeme dát WWW server na každou školu – nic z toho není špatné. Špatné je to až v tom okamžiku, když si začneme myslet, že jsme udělali něco pro vyřešení problémů se vzděláváním*.“ [21, s. 14]

Výše citovaná slova Steve Jobse byla použita jako motto této práce. Jobs ovšem, již v poněkud absurdním duchu, pokračuje: „[Abraham] *Lincoln*[11] *neměl ve srubu, kde ho jeho*

[9] Dalším absolventem školy Montessori je např. Will Wright, vývojář jedné z nejprodávanějších počítačových her *The Sims*.
[10] „*Maria Montessori začala svou pedagogickou dráhu prací s dětmi, které trpěly mentálním postižením. Docílila toho, že tyto děti složily zkoušku ze čtení a psaní určenou pro děti bez postižení. To ji podnítilo k tomu, aby svou metodu uplatnila i ve vzdělávání zdravých dětí*.“ [19, přebal] Toto hledání inspirace mj. v Montessori pedagogice si vyžaduje např. i skutečnost, že v posledních letech přibývá dětí s poruchami učení.
[11] Abraham Lincoln (1809-1865), prezident USA v letech 1861-1865

rodiče vyučovali, žádný internet, a vyrostla z něj velmi zajímavá osobnost. Historická zkušenost ukazuje, že můžeme vychovat skvělého člověka i bez technologií.“[12] A zároveň dodává, že „*… zkušenost rovněž ukazuje, že i s pomocí technologií můžeme vychovat velmi nezajímavé osobnosti.*“[13] [22]

I v odborné literatuře se setkáme s názorem, že méně je někdy více. „*…žákovi musíme dát méně, aby dosáhl větších úspěchů*.“ [1, s. 112] Jiná verze tohoto paradoxního principu říká: „*…teorii se naučíme lépe, jestliže ji méně studujeme a více uplatňujeme v praxi. Tak například je třeba předkládat méně textů ke čtení a více úkolů k praktickému provádění, abychom mohli opravovat obvyklé omyly, které žák při učení dělá. Žák je totiž většinou naprosto zahlcen, jestliže mu fungování programu vysvětlujeme s příliš mnoha doporučeními, vysvětleními, pravidly, postupy, shrnutími, popisy atd*.“ [1, s. 112]

Také autoři novozélandského projektu CSunplugged[14] - projektu výuky informatiky pro děti na základní škole - konstatují: „*Zjistili jsme, že mnoho důležitých konceptů lze vyučovat bez počítače. Dokonce je někdy počítač při učení jen rušivým prvkem*.“[15] [23, s. i] Rovněž starší původní českojazyčná odborná literatura tvrdí, že „*rozumně chápaná ‚počítačová gramotnost‘ nemá … nic společného s ‚počítačovou horečkou‘, alespoň ne v jejích hysterických formách, kdy se rodiče předhánějí v tom, kdo koupí své ratolesti modernější počítač a shánějí jim kondice z programování v domnění, že jinak se jejich dítka ve společnosti neuplatní.* [24, s. 10]

Většina odborné veřejnosti ale volá po vybavení škol nejmodernější technikou. „*Není … možné, aby děti chodily do školy jako do muzea. Je potřeba, aby učitelé byli vybaveni alespoň stejně jako jejich žáci a studenti*.“ [25] To ale vzhledem k aktuální hospodářské a finanční situaci zřejmě nebude možné. „*Podíl rozpočtu ministerstva školství na státním hrubém domácím produktu se snižuje a v porovnání s ostatními zeměmi Evropské unie se odklání od průměru směrem dolů. Vyhlídky do dalších let příliš nadějí nebudí*.“ [26]

---

[12] „*Lincoln did not have a Web site at the log cabin where his parents home-schooled him, and he turned out pretty interesting. Historical precedent shows that we can turn out amazing human beings without technology.*“

[13] “*Precedent also shows that we can turn out very uninteresting human with technology.*”

[14] CS – computer science

[15] „*We have found that many important concepts can be taught without using a computer—in fact, sometimes the computer is just a distraction from learning*.”

Projekt PRIM (Podpora rozvoje informatického myšlení), jehož se účastní všechny pedagogické fakulty v České republice a Národní ústav pro vzdělávání, jehož cílem je podporovat změnu orientace školského předmětu informatika z uživatelského ovládání výpočetních technologií směrem k základům informatiky jako oboru, připravil pro základní vzdělávání čtyři modelové ŠVP podle stupně vybavenosti škol. Základní úroveň ŠVP, Opatrně vpřed, je určena školám, které budou přecházet na novou informatiku bez nákupu jakýchkoliv pomůcek či učebnic. Tento příklad potvrzuje, že i bez speciálního vybavení lze naplnit požadavky RVP a vybavit žáky základními kompetencemi z oblasti informatiky potřebnými pro další, středoškolské, studium. [27]

2.6 Sbírky příkladů pro výuku informatiky

Nabídka klasických tištěných papírových učebnic informatiky, zvláště pak se zaměřením na algoritmizaci a programování, nebyla nikdy nijak široká. V posledních letech se, jak již bylo řečeno v podkapitole 2.5, těžiště tvorby učebních materiálů (nejen pro výuku informatiky) stále více přesouvá do online prostředí s interaktivními výukovými aplikacemi. Přesto je několik starších papírových učebnic stále dostupných, přinejmenším v knihovnách.

2.6.1 Klasické tištěné učebnice

Z aktuálnějších materiálů, z nichž tato práce čerpá, lze zmínit pro výuku základů algoritmizace na 1. stupni ZŠ zejména skripta *CS unplugged* (Christchurch, 2015) novozélandských autorů koncepce výuky základů informatiky bez počítače. Skripta vydaná vlastním nákladem (dostupná i online) se zabývají mj. tématy jako dvojková soustava, třídicí algoritmy, algoritmizace, kryptografie apod., vše samozřejmě na úrovni žáků I. stupně základní školy. Z toho materiálu je čerpáno praktické části v 7. a 8. kapitole a podkapitole 9.1.

Pro 2. stupeň je (opět i online) dostupná učebnice *Základy informatiky pro 2. stupeň ZŠ* (Liberec, Technická univerzita, 2020) autorů Jana Berkiho a Jindry Drábkové. Tato učebnice vychází ze stejné koncepce výuky informatiky bez počítače jako výše zmiňovaná novozélandská koncepce. Obsahuje 3 hlavní kapitoly: Kódování, Modely a Systémy a technologie. Seznamuje žáky s tématy jako šifrování, kódování, komprese, binární čísla, ale např. také třeba orientované grafy. Tato učebnice byla inspirací pro úlohu popsanou v podkapitole 9.3.

Programátorská cvičebnice Radka Pelánka (Brno, Computer Press, 2012) je určena spíše pro střední a vysoké školy. (Zabývá se i takovými tématy jako např. Mandelbrotova množina nebo konvexní obal.) Přesto, ačkoliv se jedná o cvičebnici, je z ní v této práci čerpáno v některých teoretických pasážích.

Kniha *Programování pro úplné začátečníky* Radka Hylmara (Brno, Computer Press, 2009) rovněž (ačkoliv by se tak podle názvu mohlo zdát) není určena pro děti, ale spíše pro dospělé samouky. Zabývá se i takovými tématy jako abstraktní datové typy zásobník a fronta a ukazateli. Vykládaná látka je prakticky demonstrovaná na zdrojových kódech v programovacím jazyku Pascal. Používání tohoto programovacího jazyka, který, přestože jej švýcarský informatik Niklaus Wirth právě za tímto účelem v roce 1970 vytvořil [16, s. 24 a 144], dnes není trendem při výuce základů programování.[16] Ovšem i z tohoto materiálu čerpá tato práce některé zajímavé nejen teoretické citace.

Z ještě starších publikací stojí za zmínku kniha autorů Januše Drózda a Rudolfa Kryla *Začínáme s programováním* (Praha, Grada, 1992). Rovněž tato učebnice demonstruje praktické příklady na zdrojovém kódu napsaném v programovacím jazyku Pascal a stejně jako v předchozím případě i tato učebnice obsahuje některé stále platné teoretické pasáže, z nichž některé byly citovány v této práci.

Učebnice autorů Andreje Blaha a Ivana Kalaše *Imagine Logo: Učebnice programování pro děti*. (Brno, Computer Press, 2006) pracuje, jak již název napovídá, s dnes již jen okrajově, pokud vůbec, používaným programovacím jazykem Imagine Logo. Nicméně i tento starší materiál je (neustále) inspirativní a to nejen svojí teoretickou částí.

Tato práce čerpala i ze starší (online dostupné) literatury ze Spojených států amerických, konkrétně z knihy Sally G. Larsenové *Computers for Kids* (New Jersey, Creative Computing Press, 1981) a z článku Edwarda H. Carlsona *Teach Your Kids Programming* (uveřejněném v roce 1983 v dubnovém čísle časopisu Creative Computing).

[16] Problém zastaralé grafiky původního vývojového prostředí, která bývá Pascalu vyčítána, lze řešit použitím webové stránky onlinegdb.com s moderním grafickým prostředím, kde navíc Pascal, původně překládaný (kompilovaný) jazyk, funguje v interaktivním režimu jako interpretovaný jazyk (tj. bez nutnosti překladu).

2.6.2 Elektronické online zdroje

V českojazyčném prostředí patří k hlavním webovým stránkám s interaktivními aplikacemi pro výuku informatiky stránky umimeinformatiku.cz a ibobr.cz.

Stránky umimeinformatiku.cz jsou součástí projektu umimeto.cz, který poskytuje interaktivní online aplikace i pro další předměty (čeština, matematika, angličtina, fyzika, biologie). Rozsáhlá sekce umimeinformatiku.cz nabízí témata

- Algoritmické myšlení
- Logika a řešení problémů
- Kódování a modelování
- Práce s daty
- Příprava dokumentů
- Digitální technologie
- Použití technologií
- Programovací jazyk Python
- Umělá inteligence a strojové učení

Každé téma obsahuje desítky úloh různé obtížnosti na procvičení. Sekce *Tvorba programu* nabízí želví grafiku, v níž je možné sestavit zdrojový kód z puzzle dílků, podobně jako je tomu u Scratche, nebo vypisovanými příkazy jako v Pythonu.[17] K úlohám není uvedeno řešení, pouze někdy méně, někdy více dostatečná nápověda. Mezi pro děti nejzajímavější činnosti patří kreslení fraktálů,[18] proto bylo na závět této práce vybráno 5 závěrečných těžších úloh pro vykreslování fraktálů pomocí želví grafiky označených jako *Záludné* a k nim byl doplněn komentář, který by měl nalezení správného řešení usnadnit.

Webové stránky ibobr.cz jsou součástí mezinárodního projektu informatické soutěže pro žáky základních a středních škol *Bebras*[19] - *Bobřík informatiky*. [28] „*Myšlenka o Bobříkovi se zrodila v Litvě u prof. Valentiny Dagieneové z Vilniuské univerzity. ... Tento nápad jí proběhl hlavou během cesty po Finsku v roce 2003 a debat o tom, jak přitáhnout žáky ke studiu informatiky. ... Hned na začátku se k Bobříkovi připojilo několik evropských zemí:*

[17] O „vývojových prostředích Želví grafika a Scratch“ bude pojednáno konkrétní ukázkou v 5. kapitole.
[18] blíže k tématu fraktál viz podkapitola 5.4
[19] *bebras* znamená v litevštině bobr

Estonsko, Německo, Nizozemí a Polsko byly mezi prvními v roce 2006. … Česká republika se poprvé zúčastnila soutěže v roce 2008.“[20] [29]

Soutěž je určena pro žáky 3. – 9. tříd základních škol a studenty středních škol. Od roku 2023 jsou soutěžící rozděleni do kategorií podle toho, jaký ročník školy navštěvují. [30]

Tabulka 1. Věkové kategorie Bobříka informatiky [30]

Kategorie	Věk	Počet otázek	Doba testu
Mini	3. – 4. r. ZŠ	12	30 min
Benjamin	5. – 6. r. ZŠ (prima osmiletého gymnázia)	12	40 min
Kadet	7. – 8. r. ZŠ (sekunda - tercie osmiletého gymnázia)	12	40 min
Junior	9. r. ZŠ – 1. r. SŠ (kvarta – kvinta osmiletého gymnázia)	12	40 min
Senior	2. – 4. r. SŠ (sexta – oktáva osmiletého gymnázia)	12	40 min

V Archivu na webových stránkách soutěže jsou k dispozici všechny testy ve všech věkových kategoriích. Na začátku byly pouze tři kategorie, dnes je jich dvojnásobek. K dispozici je tak okolo 800 úloh, k nimž je volně přístupné nejen samotné řešení, ale i výklad k němu a vysvětlení, co má daná úloha společného s informatikou. Úlohy jsou kromě informatiky (algoritmizace, hledání nejkratší cesty, teorie grafů, šifrování) také např. z oblasti kombinatoriky a logického a abstraktního myšlení.

Webové stránky *Blockly Games – Hry budoucích programátorů* jsou sice v českojazyčné mutaci, ale některá hlášení se objevují v angličtině. Stránka nabízí několik sekcí s různými cvičeními.

- Skládačka
- Bludiště
- Pták
- Želva
- Film
- Hudba
- Rybník

[20] „*The idea of Bebras was born in Lithuania, by Prof. Valentina Dagiene from University of Vilnius. … The thought rushed into head during the travel around Finland in 2003 and discussions about how we could attract pupils to learn informatics. … Since its beginnings, several European countries have joined Bebras: Estonia, Germany, The Netherlands, and Poland were the firsts to join in 2006. … Czech Republic … started their challenges in 2008*.“

Zatímco úvodní Skládačka obnáší všeho všudy jedno jednoduché cvičení na kompletování obrázků zvířat s počtem nohou a vlastnostmi (zřejmě šlo tvůrcům o to, aby se žák naučil pracovat s bloky v podobě puzzle dílků), další okruhy již obsahují 9 nebo 10 úloh odstupňovaných podle náročnosti (v některých případech je 10. úloha jen příležitostí pro vlastní volnou tvorbu v duchu předchozích cvičení). Náročnost úloh bohužel není rovnoměrně odstupňována. V některých sekcích začínají být úlohy velmi obtížné již počínaje druhou polovinou. Některé úlohy budou podrobněji rozebrány v praktické části této práce.

Webové stránky *Microsoft MakeCode* nabízí 3 základní sekce: Arcade, která obsahuje více než 100 jednoduchých her v retro stylu, které si může uživatel upravovat. Druhá sekce se jmenuje *micro:bit.* Zde se může žák naučit jednoduchým základům programování mikropočítačů a konečně třetí sekci tvoří *Minecraft* – jedna z nejznámějších edukačních počítačových her. Ke všemu je k dispozici videotutoriál s návodem.

Ve výčtu online aplikací pro výuku informatiky, zvláště pak programování, nelze vynechat programovací jazyk Scratch, pro který je typické skládání programu z puzzle dílků, takže žáci nemají problémy s nefunkčností programu v případě překlepu v zápisu zdrojového kódu (jako např. dříve při používání Pascalu) nebo špatného odsazení (v případě Pythonu). Ukázka práce s želví grafikou v Pythonu a Scratchi je obsažena v 5. kapitole. Toto řešení, tolik typické pro Scratch, inspirovalo nespočet vývojářů aplikací určených pro výuku programování a algoritmizace.

3 ALGORITMIZACE

Program, který programátor vytvoří, se chová podle do něj vloženého algoritmu. [16, s. 32] Synonymem pro tvorbu programu, je psaní *zdrojového kódu* (source code). Souvislost mezi programováním a (tvůrčím) psaním je ovšem hlubší. „*Psaní je pro školu typická činnost, která formuje i schopnosti pro programování. Napsání jednostránkové slohové práce např. na téma ‚Proč mám rád jaro' obnáší uspořádání si myšlenek na několika úrovních, od nadpisu dolů po poslední odstavec a na úrovni jednotlivých vět. V neposlední řadě je potřeba si také hlídat pravopis, gramatiku a interpunkci. Toto je velmi blízká analogie k programování*."[21] [31, s. 172]

Pojem algoritmus byl již výstižně definován v závěru podkapitoly 2.3 v souvislosti s algoritmickým přístupem behavioristů k vyučování. S pojmem algoritmus se ale běžně setkáme především v informatice. Při programování je totiž „…*tvorba algoritmu … stejně důležitá jako samotné psaní kódu, ne-li ještě důležitější, … vzhledem k tomu, že k promýšlení algoritmu nepotřebujeme počítač, zabývají se lidé algoritmy odpradávna. Už ve starém Řecku a ještě mnohem dříve kreslili učenci do písku schémata algoritmů. I dnes se vědci na univerzitách zabývají často pouze algoritmy a jejich složitostí, aniž by pro ně bylo cílem psaní konkrétních programů*." [16, s. 235]

Pojem algoritmus je prekonceptuálně[22] vnímán jako postup při řešení matematických úloh a problémů. „*Algoritmy se však neomezují pouze na samotnou matematiku. Pečete-li chleba podle receptu, postupujete podle algoritmu. Pletete-li svetr podle vzoru, postupujete podle algoritmu. Naostříte-li hranu pazourku pomocí řady přesných úderů kusem parohu, což je zásadní krok při výrobě kamenných nástroji, postupujete podle algoritmu. Algoritmy jsou součásti lidských technologií už od doby kamenné*." [32, s. 12]

Tehdy se jim tak ovšem neříkalo. „*Pojem algoritmus je novodobý, byl zaveden až začátkem 20. století při studiu ‚mechanické' řešitelnosti matematických úloh*." [33, s. 30] Čili např. slovní spojení *Euklidův algoritmus* je označení *ante litteram*, tj. než se tak začalo říkat.

---

[21] „*Writing is a familiar school activity that forms a model for programming. Writing a one page theme. on ‚What I Like About Springtime' involves organizing your thoughts on several scales, from the topic as a whole down to paragraph and sentence levels. Finally, spelling, grammar and punctuation must be correct. The analogy to writing a Basic program is very close*."

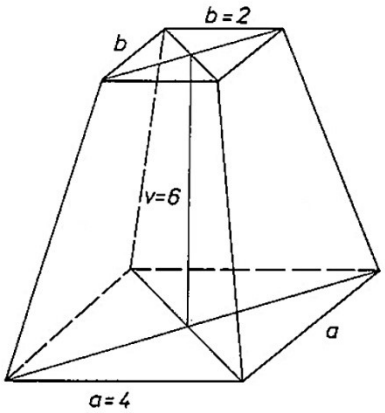
[22] Zde je již termínu *prekoncept* použito striktně v souladu s definicí uvedené v závěru úvodní kapitoly.

3.1 Mezioborový přesah

Pohled do historie algoritmů může u žáků, kteří se zajímají o moderní technologie, probudit alespoň částečný zájem o historii, a naopak žáky se zájmem o historii může inspirovat k zamyšlení o netušených souvislostech mezi historickými objevy z oblasti exaktních věd a pozadím fungování moderních technologií.[23]

„*Rozšířenou mylnou představu o lidech* [žijících v období starověku]*, vycházející z přeceňování inteligence dnešních lidí a z podceňování inteligence našich předků, dost výrazně koriguje nález z roku 1858. Je dnes znám pod názvem Rhindův papyrus (přibližně 1650 př. n. l.). Z úlohy číslo 79 tohoto papyru např. plyne, že už tehdy lidé věděli, jak počítat objem pravidelného čtyřbokého komolého jehlanu. Text úlohy doprovázíme pro názornost obrázkem …*“ [34, s. 12]

„*… překlad původního znění: ‚Způsob výpočtu komolého jehlanu. Máš-li dán komolý jehlan o výšce šesti loktů, mající dolní stranu čtyři lokte a horní stranu dva lokte dlouhou, počítej s touto 4 tak, že ji umocníš na druhou. Získává se 16. Zdvojnásob 4; získává se 8. počítej s touto 2 tak, že ji umocníš na druhou. Získává se 4. Sečti těchto 16 s těmito 8 a touto 4. Získává se 28. Obsáhni 3 v 6. Získává se 2. Počítej 28 dvakrát. Získává se 56. Pohleď: On je 56. Našel jsi správně.‘ Číselná formulace, pomocí které je vykládáno řešení, je typická i pro ostatní úlohy dochované z té doby*.“ [34, s. 12-13]



Obrázek 4. Komolý jehlan [34, s. 12]

[23] Mezipředmětové *přesahy*, nebo také mezipředmětové *vztahy* jsou (rovněž) jedním z prvků, na které klade moderní pedagogika důraz.

Německý archeolog Heinrich Schliemann (1822-1890), objevitel měst Trója a Mykény, které byly centrem první řecké civilizace, zastával názor, že „… *nemáme říkat ‚už', mluvíme-li o starověku. Věta ‚už tenkrát měli vysoce vyspělou techniku' vyznívá totiž v tom smyslu, že lidé ve starověku měli jen nepatrné zlomky našich vědomostí. Ve skutečnosti je tomu naopak: my tyjeme z pozůstatků jejich znalostí a dosud jsme odhalili jen velmi málo z toho, co uměli a jak to dělali*." [35, s. 152]

Schliemann, který se zabýval starořeckou kulturou, mohl snad něco podobného tvrdit. O Egypťanech a Babyloňanech podobné tvrzení, přinejmenším v souvislosti s matematikou, neplatí. „*Přes 4000 let nebylo dosaženo v této oblasti* [v oblasti matematiky] *téměř žádného pokroku. A navíc ještě nebyla ani objevena podstata matematiky, totiž uvažování vedoucí ke stanovení hodnoty metody a výsledku*." [36, s. 14]

Naproti tomu dílo *Základy* řeckého matematika Euklida (asi 325 př. n. l. – asi 260 př. n. l.), „*… který kolem roku 300 př. n. l. sestavil a systematizoval vše, co bylo v jeho době známo o planimetrii a stereometrii, … bylo tak důkladné a spolehlivé, že se stalo v podstatě biblí geometrie po více než dva tisíce příštích let a tím i jedním z nejtrvanlivějších děl všech dob*." [37, s. 112] … „*Z dlouhodobého pohledu byl Eukleidův nedostatek absolutní přesnosti po více než dvou tisících let od sepsání jeho díla příčinou jednoho z nejplodnějších průlomů v matematice vůbec*." [37, s. 113][24]

3.2 Aktuálnost a praktičnost

Jeden z největších matematiků 20. století, Godfrey Harold Hardy (1877-1947), tvrdí, že „*…jen velmi málo matematiky je použitelné v praxi a i to málo je poměrně nezajímavé*." [38, s. 81-82] Na jiném místě ve své knize *Obrana matematikova* (*A Mathematician's Apology*, 1940) Hardy píše: „*Říká se občas, že čistí matematici jsou hrdí na neužitečnost své práce a holedbají se tím, že nemá žádné praktické použití. Obvinění se obvykle zakládá na*

[24] Citovaná kniha na pomezí metamatematiky, informatiky, filozofie, kognitivní psychologie a dalších oborů, např. muzikologie a kunsthistorie, získala v roce 1980 Pulitzerovu cenu v kategorii vědecké literatury. Její autor, Douglas Hofstadter (*1945), je synem fyzika Roberta Hofstadtera (1915-1990), nositele Nobelovy ceny za fyziku z roku 1961. Kniha, zejména při hledání analogií s hudbou, obsahuje několik i hrubých nepřesností. Nejvýraznější je tvrzení, že Dvořák byl hluchý. Americký autor si spletl Antonína Dvořáka s Bedřichem Smetanou. Tato chyba se objevuje i v českém překladu (s. 163) bez jakékoliv poznámky (pod čarou), která by tento omyl uvedla na pravou míru.

neopatrném výroku připisovaném Gaussovi, že je-li matematika královnou věd, pak je teorie čísel, díky své naprosté zbytečnosti, královnou matematiky …“ [38, s. 109]

Mnohé objevy se v době, kdy byly učiněny, zdály být neužitečné a jejich praktičnost se ukázala až časem. „*Čas může všechno změnit. Nikdo nepředvídal využití matic, grup a jiných matematických teorií v moderní fyzice. Stává se, že se nějaká ‚intelektuální‘ aplikovaná matematika stane neočekávaně ‚užitečná‘, ale dosavadní zkušenost potvrzuje závěr, že jak v jednom, tak ve druhém oboru je to právě to všední a nezajímavé, co má nějaký význam pro praktický život*.“ [38, s. 119]

„*V každé vědě znamenají nejdůležitější pokrok takové objevy a zobecnění, které zabezpečují předstih celého vědního úseku na dlouhou dobu. Ty se stávají základními kameny ve stavbě celé vědní disciplíny. Jejich vliv je všeobecný a rozhodující i v případě, nemají-li ihned aplikace v praxi, např. v technice. Tyto výsledky, i když patří do „čistě" matematiky, představují mocný arzenál nejrozmanitějších poznatků, který je stále připraven k použití dnes nebo později při řešení konkrétních otázek moderní techniky. Příkladem toho je teorie kuželoseček, kterou v klasickém Řecku vybudoval Apollonios z Pergy (265-170 před n. l.) a která se až po 18 stoletích uplatnila v Keplerových zákonech pohybu nebeských těles. Podobně je tomu s některými zcela abstraktními výsledky matematické logiky a abstraktní algebry, jichž bylo znamenitým způsobem použito v automatizaci, telemechanice a kybernetice*.“ [39, s. 30]

Mezi výše zmíněné kuželosečky patří i elipsy, u nichž lze najít souvislost s informatikou, konkrétně s kybernetickou bezpečností. Samotné elipsy toho sice se šifrováním mnoho společného nemají, „… *ale od nich je jen krůček k eliptickým integrálům, funkcím a křivkám. Matematikové si s těmito pojmy hráli tak dlouho, až, ostatně jako obvykle, z toho vyšlo něco praktického. Z čisté algebry vzniká po více než sto letech výzkumu „do šuplíku” v roce 1985 (pracemi V. Millera a N. Koblitze) zcela nová perspektivní oblast moderní aplikované kryptografie. Kdyby tak zakladatelé tohoto směru věděli, že se jejich myšlenky dostanou až na jakési bezkontaktní čipové karty a do čipů pro jakési technologie GSM! V současné době pronikly eliptické kryptosystémy do řady světových standardů a staly se alternativou ke ‚klasickému’ RSA i DSA. Mají své výhody zejména v rychlosti a menší náročnosti na hardware i software*.“ [40, s. 134]

„*Matematikové tak nejenže předjímali potřeby vědy, ale také určovali směr, kterým se má věda vydat*.“ [41, s. 473]

3.3 Vlastnosti algoritmu

Algoritmus musí mít následující vlastnosti:

- Konečnost

Program pracující s daným algoritmem musí skončit po určitém počtu provedených kroků a nezacyklit se. Stejně jako v mnoha jiných oborech, také v informatice (občas) platí, že výjimka potvrzuje pravidlo. „*Některé počítačové programy jsou paradoxně založeny na tom, že nikdy nekončí. Ovšem jsou to speciální případy. Nejznámějším z nich je operační systém. Když zapnete počítač, načte se systém, který nemá žádnou ukončovací podmínku. Pokud sám uživatel počítač nevypne, běží systém skutečně donekonečna*.“ [16, s. 34]

- Jednoznačnost

„*Jednoznačnost* [algoritmu] *zaručuje, že v každém kroku, v každém stavu algoritmu je jasně dané, jaká činnost se v daném stavu provádí a jak pokračovat dál, tedy do kterého stavu a s jakými informacemi se posunout*.“ [16, s. 34]

- Nutnost výstupu

„*Algoritmus musí vždy vrátit nějaký výsledek. … ať už je výsledkem cokoli, algoritmus by nám to měl dát vědět. Třeba i chybovým hlášením*.“ [16, s. 35]

- Obecnost

Nebo také univerzálnost. Algoritmus musí být „…*vhodný pro celou třídu obdobných problémů …musí zkrátka fungovat pro všechny možnosti vstupu*.“ [16, s. 34]

„*Kromě těchto čtyř základních vlastností se někdy uvádějí některé další. Jedním z požadavků, které se kladou na algoritmus, je správnost výsledku. … Požadavek správnosti už* [ale] *nepatří mezi základní, protože nesprávnost výsledku by mohla být záměrem programátora*.“ [16, s. 35]

Navíc dokázat, že algoritmus je správný, je značně obtížnější než dokázat, že není správný. [24, s. 44] “*Nikdy nestačí provést konečný počet výpočtů k tomu, abychom ukázali, že algoritmus je správný. … zkušební provádění výpočtů algoritmu není nikdy pokusem o důkaz jeho správnosti, ale naopak snahou ukázat, že algoritmus správný není!*“ [24, s. 44]

Další vlastností algoritmu, která se někdy uvádí jako jedna ze základních, je složitost. [16, s. 35] Následující konkrétní příklad demonstruje problém univerzálnosti algoritmu.

3.4 Pascalův trojúhelník

Francouzský matematik Blaise Pascal (1623-1662) se zabýval mj. hazardními hrami a stal se jedním ze zakladatelů teorie pravděpodobnosti. Sestavil trojúhelník, který nese jeho jméno, Pascalův trojúhelník, z něhož se vychází při určování pravděpodobnosti. [42, s. 149] Na vrcholu trojúhelníku i na okrajích každého řádku je vždy jednička. Zbylá čísla jsou vždy součtem hodnot dvou sousedních čísel umístěných ve vyšším řádku vlevo a vpravo nad daným číslem. [43, s. 73]

					1						0
				1		1					I
			1		2		1				II
		1		3		3		1			III
	1		4		6		4		1		IV
1		5		10		10		5		1	V

Obrázek 5. Pascalův trojúhelník [42, s. 149]

Do páté úrovně, včetně nulté,[25] odpovídají čísla v daném řádku hodnotě příslušné mocniny čísla 11, konkrétně $11^0 = 1$, $11^1 = 11$, $11^2 = 121$, $11^3 = 1331$, $11^4 = 14641$ až hodnota $11^5 = 161051$ nevytváří palindromické číslo.[26] [44, s. 52] Algoritmus, viz Obrázek 6, který při sestavování Pascalova trojúhelníku pracuje s druhou odmocninou, tak nesplňuje požadavek obecnosti, tj. univerzálního algoritmu, který vrátí správné řešení při zadání libovolného vstupu.

[25] Naučit žáky pracovat s nultou mocninou a počítat od nuly je nepochybně dobrým vkladem do dalšího studia informatiky, zejména pro práci s cykly a indexování.

[26] palindrom – text, který se čte z obou stran stejně
Přesnější by ale byla formulace, že zápis hodnoty 161 051 v dekadické soustavě nevytváří palindrom. Např. číslo 27 také není palindrom v dekadické soustavě, ale zápis odpovídající hodnoty v binární soustavě, 11011, palindromem je.

1^0	1	✓
11^1	1 1	✓
11^2	1 2 1	✓
11^3	1 3 3 1	✓
11^4	1 4 6 4 1	✓
11^5	1 5 10 10 5 1	✗

Obrázek 6. Pascalův trojúhelník vytvořený pomocí neuniverzálního algoritmu

Informatická literatura zpravidla uvádí jako příklad algoritmu, jež nesplňuje požadavek obecnosti, Fermatův vzorec pro generování prvočísel. [24, s. 21 a 43-44] Ten je ovšem mnohem složitější a pro žáka základní školy obtížnější na pochopení a tudíž spíše nevhodný.

3.5 Algoritmizace a Bloomova taxonomie

„*Algoritmus* [lze chápat jako zápis] … *postupu, použitelného pro řešení určité třídy problémů. ... Jestliže již řešení známe, potřebujeme je zapsat jako algoritmus. Přitom postupujeme obvykle tak, že postup řešení rozkládáme na jednodušší operace, až dospějeme k elementárním krokům. Tento postup návrhu algoritmu se obvykle označuje jako metoda shora dolů*.“ [45, s. 10]

Postupem shora dolů (*top-down design*) vznikají často tzv. rekurzivní algoritmy. „*Někdy je ovšem nezbytné transformovat algoritmus do nerekurzivní podoby – např. proto, že použitý programovací jazyk ji nedovoluje*.“ [46, s. 60]

Postup metodou shora dolů odpovídá 4. bodu Bloomovy taxonomie kognitivních cílů, který hovoří o schopnosti rozdělit informaci, v našem případě problém, na menší, vzájemně související části a schopnosti vytvořit logickou posloupnost.

Naproti tomu opačný postup, zdola nahoru (*bottom-up design*), odpovídá 5. bodu Bloomovy taxonomie, který hovoří o schopnosti složit informaci do originálního smysluplného celku, schopnosti kompletovat informace.

Postup zdola nahoru je vhodný pro zkušenější programátory. „… *pro začátečníky je vhodný pouze ve chvíli, kdy si pouze hrají a staví různé součásti bez přesného zadání, co má být cílem daného návrhu. Tak trochu podle hesla: ‚Ono se ukáže, k čemu je to dobré*.‘“ [17, s. 99] Zároveň je postup zdola nahoru výhodnější „… *při návrhu programů spolupracujících s některými zařízeními (např. s chytrou domácností), při němž si musíte nejprve ujasnit, co vše můžete po takovém zařízení chtít a jak toho dosáhnout* …“ [17, s. 100]

Tabulka 2 přináší přehledné srovnání obou přístupů. Lze z ní „… *odvodit, kdy kterou z metod zvolit, případně jak je zkombinovat*.“ [17, s. 99]

Tabulka 2. Porovnání metody shora dolů a zdola nahoru [17, s. 100]

	Shora dolů	Zdola nahoru
Podstatou postupu je	analýza	syntéza
Nový prvek je	prostředkem	cílem
Že nový prvek půjde nad danou základnou vytvořit	nevíme jistě, spíše předpokládáme	víme jistě
Jaké bude mít nový prvek vlastnosti	víme přesně	přizpůsobíme to možnostem dostupných knihoven
Jak bude nový prvek zkonstruován	vyřešíme později	musíme se tím zabývat hned
Konkrétní použití nového prvku	známe	nezajímá nás (může být libovolné)
Předpokládaná aplikace	v jednom projektu	v mnoha projektech
Návratnost vynaložené námahy	brzká, jednorázová	opožděná, postupná
Hlavní riziko	požadujeme něco, co nepůjde naprogramovat nebo nebude efektivní	vytvoříme něco, co se nebude dostatečně využívat

4 DATA, INFORMACE A MODELOVÁNÍ

Vedle tematického celku Algoritmizace a programování zavádí nový RVP pro informatiku také tematický celek Data, informace a modelování. Do tohoto celku spadá i kódování, které je při tvorbě (zvláště rozsáhlejších) programů a informačních systémů stejně důležité jako modelování.

4.1 Kódování

„*Kódování je metoda transformace dat z jedné standardizované reprezentace do jiné takovým způsobem, který je obecně známý. Při kódování jsou na vstupu určitá data a na výstupu jsou ty samé data zakódované pouze do jiné reprezentace, kterou určuje daný standard kódování*.“ [47] Příkladem kódu je např. čárový kód na zboží v obchodech, QR kód, různé piktogramy nebo zkratky (např. internetové domény) či různé číselné a alfanumerické kódy.

Některé číselné kódy jsou natolik sofistikované a systémové, že obsahují např. kontrolní prvek, který upozorní na možný překlep při ručním opisování kódu do různých elektronických informačních systémů. Takovým případem je třeba číselný kód ISBN (International Standard Book Number), u nějž tuto kontrolní funkci plní poslední číslice. [23, s. 38]

Například u ISBN-kódu 0-13-911991-4 je to tedy číslo 4. Kontrolní systém funguje tak, že prvních devět číslic, které plní funkci různých identifikátorů (žánr nebo obor, jakého se kniha týká, jazyk, v němž je kniha napsána, apod.) se vynásobí podle následujícího klíče: první číslice se vynásobí deseti, druhá devíti, třetí osmi atd. až po devátou číslici, která se vynásobí dvěma. Jednotlivé součiny se pak navzájem sečtou. [23, s. 38]

$$(0 \times 10) + (1 \times 9) + (3 \times 8) + (9 \times 7) + (1 \times 6) + (1 \times 5) + (9 \times 4) + (9 \times 3) + (1 \times 2) = 172$$

Výsledný součet se pak dělí 11, přičemž důležitý je zbytek po celočíselném dělení. Pokud je zbytek 0, pak je poslední, tj. kontrolní, číslo ISBN-kódu 0. V ostatních případech se zbytek odečte od 11 a posledním kontrolním číslem v ISBN-kódu je rozdíl tohoto odčítání. [23, s. 38]

172 : 11 = 15 a zbytek je 7. Následně 11 - 7 = 4. V našem případě je tedy kontrolním číslem uvedeným na konci ISBN-kódu opravdu právě číslo 4. [23, s. 38] Podobné kontrolní mechanismy fungují u všech kódů, kde by včas neodhalený překlep při psaní mohl později

způsobit velké problémy, jako např. čísla bankovních účtů, daňová identifikační čísla apod. [23, s. 39]

Kódování, přesněji řečeno číselné kódování informací, znaků a příkazů je základní podmínkou k tomu, aby mohl zpracování informací a dat[27] převzít stroj, v našem případě počítač. [48, s. 14] Základem fungování počítače je binární číselný kód, zvaný také dvojková soustava.

Ke kódování dále viz praktický příklad v podkapitole 9.3 v Praktické části této práce.

4.2 Modelování

„*Zatímco kódování se zabývá především tím jak informace zachytit, modelování se zabývá i tím, co chceme zachytit. Model je zjednodušené znázornění skutečnosti*. *Díky …* [zjednodušení] *nám ale dobrý model umožňuje o skutečnosti lépe přemýšlet*. *… s modely pracujeme neustále, i když o nich třeba nepřemýšlíme jako o modelech*.“ [49]

4.2.1 Modely

„*Typický příklad* [modelu] *je mapa. Mapa je model prostoru. Turistická mapa například zachycuje polohu míst, vzdálenosti, nadmořské výšky. Zachycuje také tvar cest, ale jen zjednodušeně. A mnohé* [pro účel, kterému má sloužit, nepotřebné] *věci ze skutečnosti v mapě vůbec nejsou.*[28] *… Kvalitu modelu neposuzujeme podle míry detailu, ale podle toho, jak plní svůj účel*.“ Chování modelu pak zachycuje simulace. [49]

[27] Gramatika českého jazyka považuje termín *data* v souvislosti s výrazem označujícím informace zpracované počítačem za podstatné jméno pomnožné, tj. slovo existující pouze v množném čísle. Nicméně odborná literatura z oblasti informatiky uvádí jako jednotné číslo od pomnožného slova *data* pojem *údaj*.

[28] Tím, že model zjednodušuje, co je složité, zmenšuje, co je velké nebo zvětšuje, co je malé, plní současně charakter učební pomůcky, viz kapitola 2.1.

Obrázek 7. Schéma stanic metra [49]

Ze schématu (modelu) stanic metra, viz Obrázek 7, nepoznáme vzdálenost mezi jednotlivými stanicemi metra, ale poznáme, které stanice jsou přestupní a můžeme si tak optimálně naplánovat cestu tak, abychom se dostali do cílového bodu s co nejmenším počtem přestupů. Barvy čar odpovídají barvám linek metra pro lepší orientaci nejen při vlastním cestování, ale už i při plánování. [49]

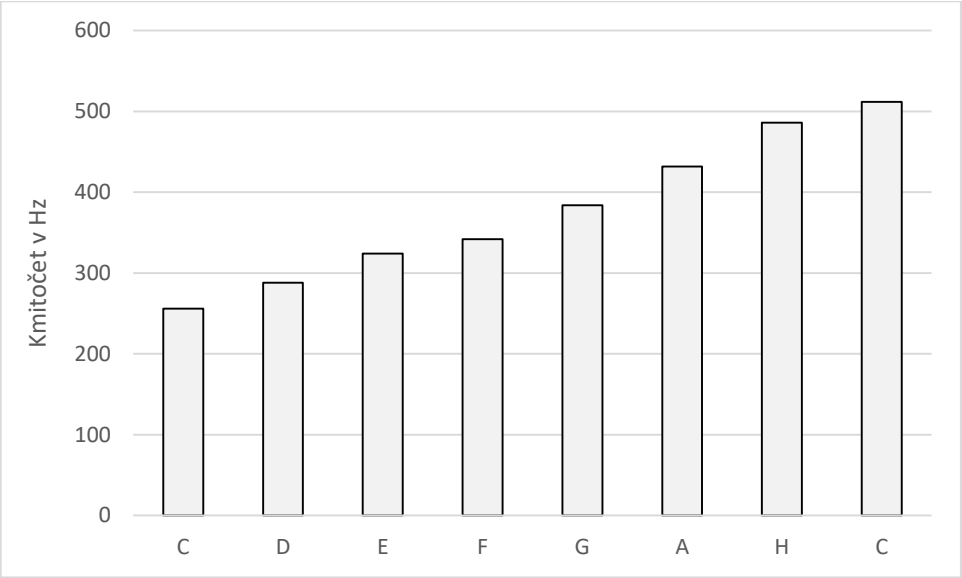
4.2.2 Grafy

Termín *graf*[29] se nejen v matematice používá ve dvou zcela odlišných významech – jednak jako graf funkce nebo nějaké grafické vyjádření poměrů hodnot, např. pomocí sloupcových nebo výsečových grafů, a jednak jako soustava bodů (tzv. uzlů nebo také vrcholů grafu), mezi nimiž vedou spojnice (tzv. hrany grafu). Takovými grafy se zabývá disciplína zvaná teorie grafů. [50, s. 13] „*Teorie grafů nám dává elegantní nástroj k popisu situací ze skutečného i matematického světa. Díky tomu dovedeme různé praktické problémy překládat na otázky týkající se grafů.*“ [s. 33, 107]

V informatice se setkáme s grafy v obou výše popsaných významech. Děti, které se teprve se základy informatiky seznamují, se zpravidla dříve setkají s prvním uvedeným typem grafu. Např. Obrázek 8 znázorňuje na svislé ose *y* hodnotu kmitočtů jednotlivých tónů ve

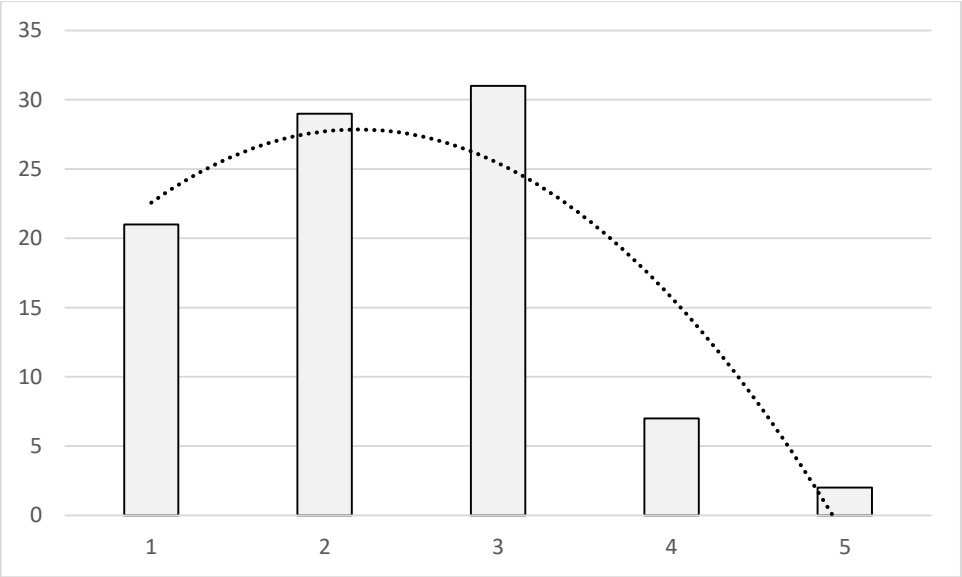
[29] Základ slova je v řeckém *grafō* – píši, za graf tak lze označit vše, co je napsané nebo nakreslené; viz také termíny jako *grafika*, *grafologie*, *grafoman*, *polygrafie*, *dysgrafie* apod.

stupnici C dur. Mezi tóny *E* a *F* a tóny *H* a *C* je opticky patrný půltónový krok oproti celotónovým krokům v ostatním případech.



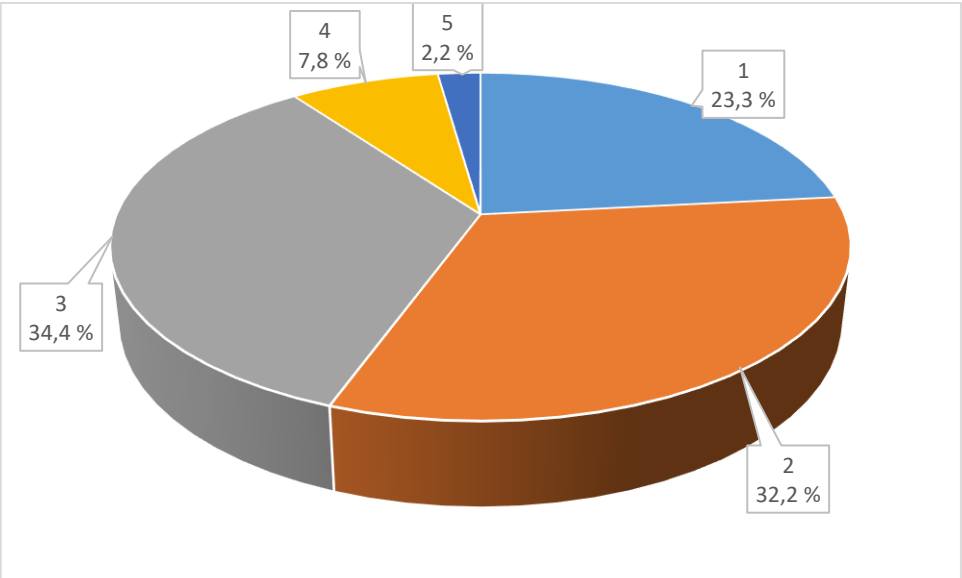
Obrázek 8. Grafické znázornění kmitočtů tónů ve stupnici C dur [43, s. 78 a vlastní]

Jiný příklad, viz Obrázek 9, graficky zobrazuje statistické vyhodnocení výsledků písemky z matematiky, kterou psal celý ročník na škole, tj. třídy A, B i C. Na vodorovné ose *x* je znázorněna známka (1 až 5) a na svislé ose *y* počet žáků, kteří příslušnou známku dostali. Spojnice trendu připomíná tzv. Gaussovu křivku přirozeného rozdělení, kdy nejčastěji jsou zastoupeny střední hodnoty a od středu směrem do stran počty postupně klesají.



Obrázek 9. Výsledky písemky z matematiky zobrazené pomocí sloupcového grafu

Uvedený příklad ale lze zpracovat, tj. graficky zobrazit, i výsečovým grafem – viz Obrázek 10, který ukazuje procentuální podíl jednotlivých známek. Schopnost samostatné správné volby grafického zobrazení zpracovávaných dat odpovídá 3. bodu Bloomovy taxonomie, který hovoří o schopnosti vhodně použít znalosti v konkrétních situacích, příp. najít řešení daného problému.



Obrázek 10. Výsečový graf vyjadřující procentuální rozložení jednotlivých známek

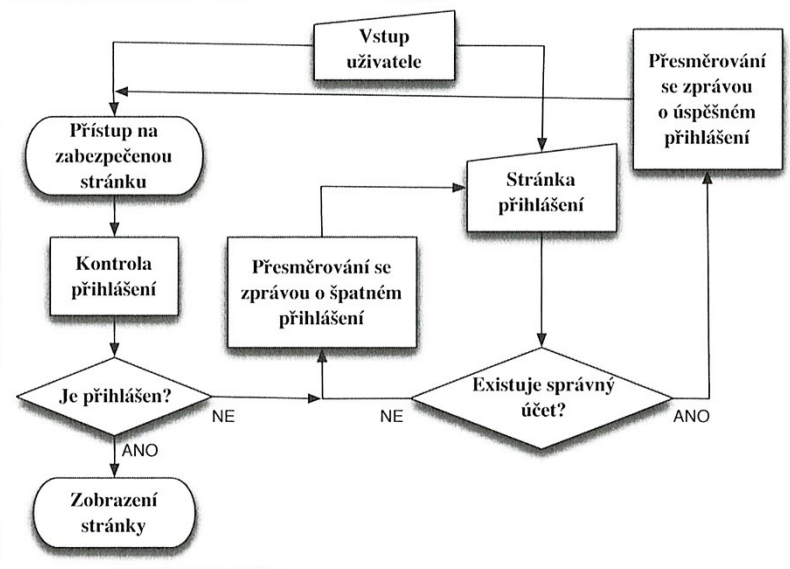
4.2.3 Vývojové diagramy

Ačkoliv se v odborné literatuře setkáme s názorem, který považuje používání vývojových diagramů pro zobrazování algoritmů sekvenčního[30] typu za nešvar [46, s. 17], patří vývojový diagram (*flowchart*) mezi nejběžnější způsoby grafického znázornění průběhu konkrétních algoritmů. Další možností vyjádření algoritmu je slovní popis, matematický zápis, např. algoritmus pro výpočet objemu komolého jehlanu popsaný v podkapitole 3.1, je asi nejvhodnější vyjádřit vzorcem

$$V = \frac{h}{3}(a^2 + ab + b^2)$$

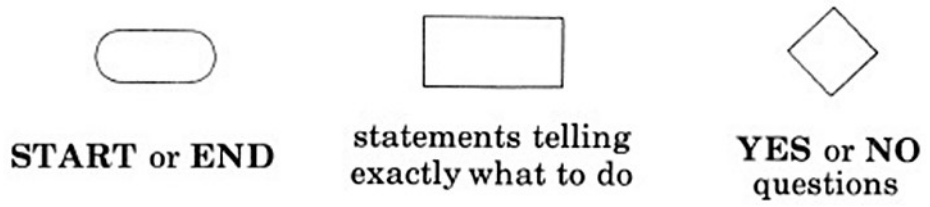
[30] sekvenční provádění znamená vykonávání jednoho příkazu za druhým

nebo zápis v tzv. pseudokódu, tj. abstraktního zápisu, „… *který je příjemně srozumitelný člověku, ale také se dá s minimem úsilí převést do libovolného programovacího jazyka*“ [33, s. 11], příp. zápis (funkčního) zdrojového kódu v konkrétním programovacím jazyku.



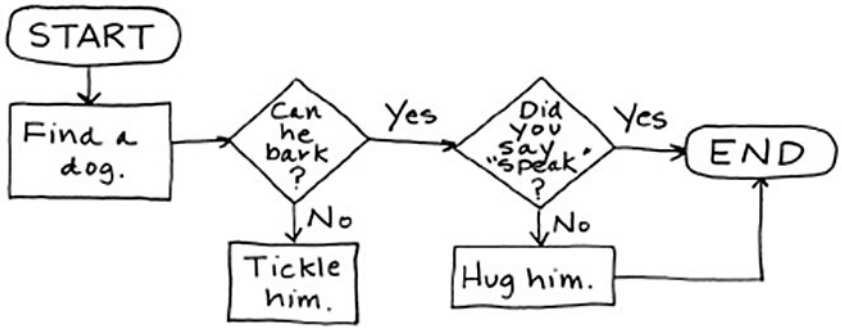
Obrázek 11. Vývojový diagram přihlášení k webové aplikaci [46, s. 17]

Nejčastěji se u vývojových diagramů setkáme se třemi tvary obrazců, které symbolizují povahu daného kroku: ovál se používá pro začátek a konec, obdélník jasné říká, co přesně se má provést (není jiná volba), a konečně kosočtverec představuje jakousi rozhodovací křižovatku, na níž se průběh algoritmu tzv. větví podle toho, zda je daná podmínka splněna nebo ne. Toto rozhodování pak mění pořadí prováděných kroků. [51, s. 4]



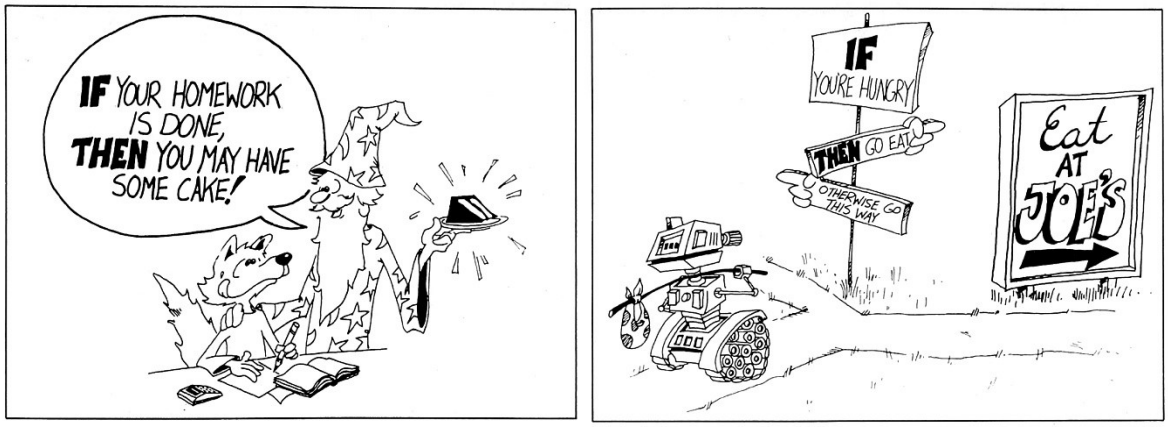
Obrázek 12. Základní symboly používané ve vývojových diagramech [51, s. 4]

Protože vývojový diagram je, jak bylo řečeno výše, grafickým vyjádřením nějakého konkrétního algoritmu, musí i u vývojového diagramu nakonec všechny kroky směřovat k ukončení. U vývojového diagramu, viz Obrázek 13, dospěje průběh, v případě, že nalezený pes neštěká, do slepé uličky (*dead end*). Ve vývojovém diagramu chybí šipka z příkazu „(po)hraj si s ním“ (*tickle him*) k ukončení celého zobrazeného procesu.



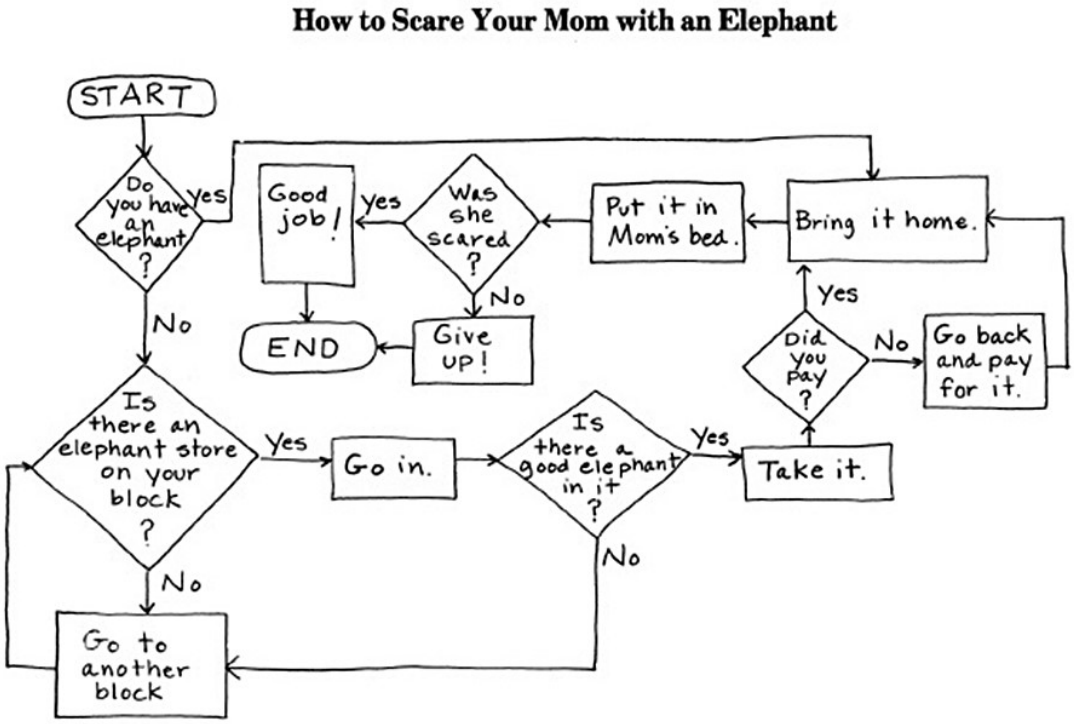
Obrázek 13. Vývojový diagram s mrtvým bodem [51, s. 6]

Část odborné veřejnosti považuje (podmíněné) větvení (*conditional branching*), tj. rozhodování, zda je daná podmínka splněna nebo ne (*If ... Then*), za pro děti obtížné na pochopení a doporučují použít k jejímu výkladu vizuální metafory jako např. [31, s. 173]



Obrázek 14. Vizuální metafory pro pochopení rozhodování [31, s. 172 a 173]

Zkušenosti ze Spojených států amerických ukazují, že učební látku obsahující vývojové diagramy s větvením lze předložit už dětem začínajícím s informatikou na 1. stupni základní školy. Obrázek 15 představuje již docela složitý vývojový diagram s názvem *Jak vylekat maminku slonem*, který vytvořila osmiletá holčička. [51, s. 5]



Obrázek 15. Vývojový diagram vytvořený osmiletým dítětem [51, s. 5]

5 ŽELVÍ GRAFIKA

Na úvod praktické výuky budoucích programátorů u počítače se nedoporučuje začínat výukou aritmetiky. „*Je to opravdu nezáživné i pro děti, kterým aritmetika nedělá potíže. Počkejte s tím, až si nějaký program bude žádat nějakou drobnou aritmetickou operaci. Pak k tomu bude vhodný čas*.“[31] [31, s. 173] Pro začátek jsou vhodnější úlohy, „… *kde výstupem jsou obrázky. Díky tomu jsou tyto úlohy atraktivní, protože většinu lidí baví pracovat s obrázky, byť jednoduchými. Studenti také bývají více motivováni vyřešit tyto úlohy správně než u úloh pracujících s čísly. Pokud program vykresluje obrázek špatně, většinou to „pobuřuje" vrozený estetický cit člověka a díky tomu má student větší motivaci program spravit. Pokud program vypisuje špatné číslo, člověka štve, že má program špatně, ale většinou nemá problém s tím, že by se mu výstup ‚nelíbil'*.“ [52, s. 43]

Za tímto výukovým účelem byla vyvinuta tzv. želví grafika, se kterou „… *přišel v roce 1969 jazyk Logo*[32] *určený pro kurzy programování. Ten se stal v 80. letech po nástupu osobních počítačů velmi populární a s ním i jeho želví grafika. Moduly se želví grafikou již dnes existují pro většinu jazyků používaných ve vstupních kurzech programování. Mimo jiné je tento modul součástí standardní instalace Pythonu.*“ [53, s. 112]

V této kapitole věnované vektorové grafice jsou uvedena řešení v jazyku Python a následně, pro srovnání, v jazyku Scratch. „*Jazyk Python byl pro ukázky zvolen proto, že jde o čistý vysokoúrovňový jazyk, který bývá občas označován za ‚spustitelný pseudokód'. Ukázky kódu by tedy měly být pochopitelné i bez znalosti tohoto konkrétního jazyka*.“ [52, s. 11]

Kromě instalovaného programu Python přímo na počítači lze pracovat s želví grafikou v Pythonu i online (a to bezplatně a bez registrace) např. ve vývojovém prostředí na webovém portále pythonsandbox.com/turtle. V tomto případě pak není nutný zápis pro import modulu *turtle*, je zde již importován.

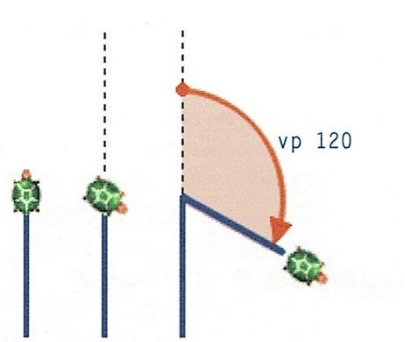
5.1 Trojúhelník a čtverec

U první úlohy si na samém začátku vystačíme s příkazy *forward* (dopředu), příp. *back* (dozadu) s uvedením vzdálenosti v přilehlé závorce a *left* (doleva), příp. *right* (doprava)

[31] „*It is deadly dull, even for children who find arithmetic easy. Wait until some program requires a little calculation, then start putting it in*.“
[32] zmínka o programovacím jazyku Logo Imagine viz kapitola 2.6.1

s uvedením stupňů otočení v přilehlé závorce. Následně si soubor nástrojů, s nimiž budeme pracovat, rozšíříme o příkaz k předem danému počtu opakování zapsaných příkazů a k podmínkám.

Nejprve bude žákům předvedeno nakreslení rovnostranného trojúhelníku bez cyklického příkazu. Žáci budou nepochybně vědět, že součet (vnitřních) úhlů u trojúhelníku je 180° a u rovnostranného trojúhelníku je to u všech tří vrcholů 60°. Možná je ale překvapí, že příkaz k otočení směru udává 120°. Zde je třeba vysvětlit, že abychom dosáhli úhlu 60°, je potřeba se z přímého směru (180°) otočit právě o onen rozdíl 180 - 60, čili oněch 120°. [54, s. 17]



Obrázek 16. Přímý úhel a vedlejší úhly [54, s. 17]

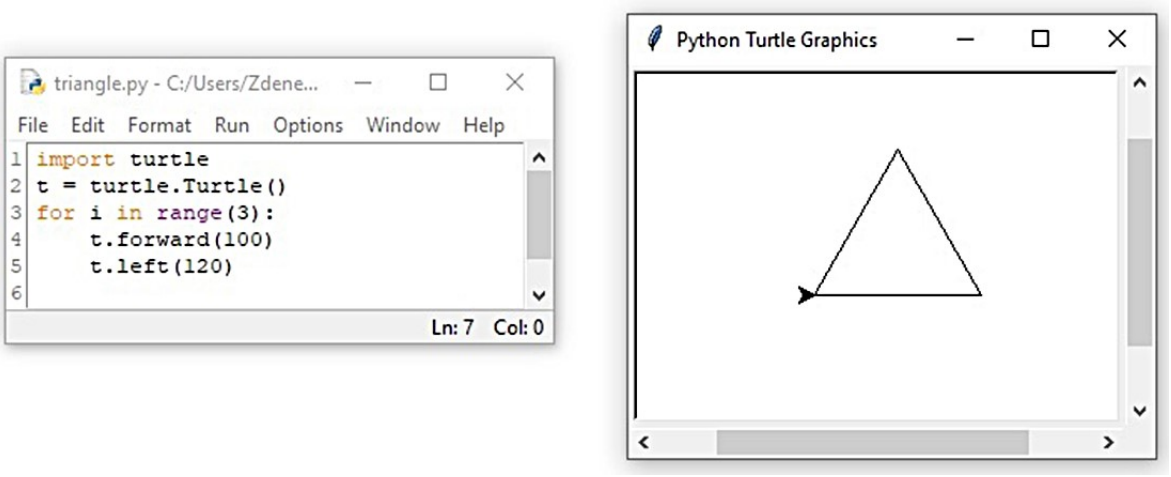
triangle.py - C:/Users/

File Edit Format Run Options Window Help

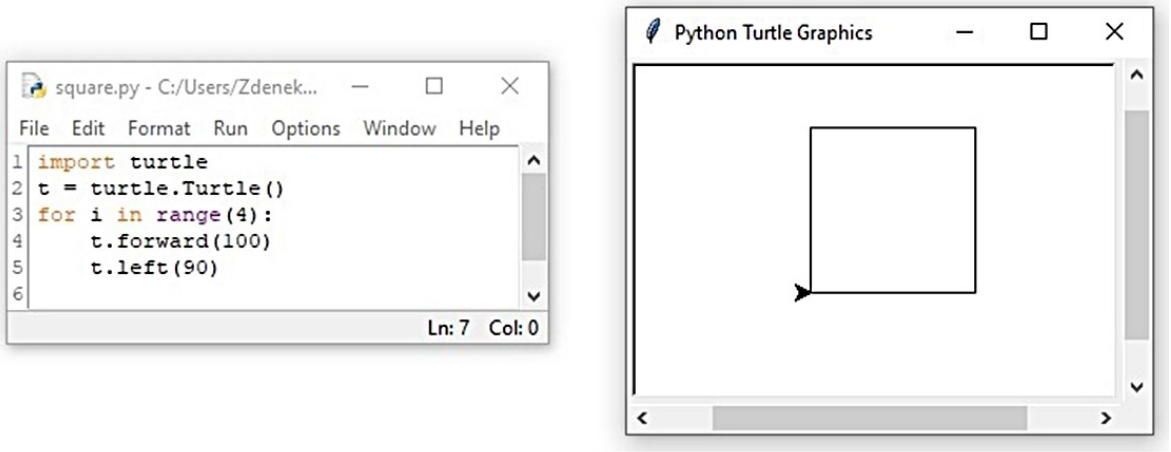
```
1 import turtle
2 t = turtle.Turtle()
3 t.forward(100)
4 t.left(120)
5 t.forward(100)
6 t.left(120)
7 t.forward(100)
8 t.left(120)
9
```

Obrázek 17. Jednoduchý zdrojový kód pro trojúhelník

Následně bude žákům s výkladem o programátorské zásadě DRY – *Don't repeat yourself* (neopakuj se), podle níž „… *by se v programu neměly vyskytovat dva shodné, nebo velmi podobné úseky kódu (nebo dokonce více)*“ [17, s. 93], ukázáno použití příkazu k vytvoření cyklu *for i in range()*, kdy v přilehlé závorce je udán počet opakování. Zároveň bude nutný i stručný výklad o odsazování a jeho důležitosti zvláště u jazyka Python.



Obrázek 18. Zdrojový kód pro trojúhelník pomocí cyklu



Obrázek 19. Zdrojový kód pro čtverec pomocí cyklu

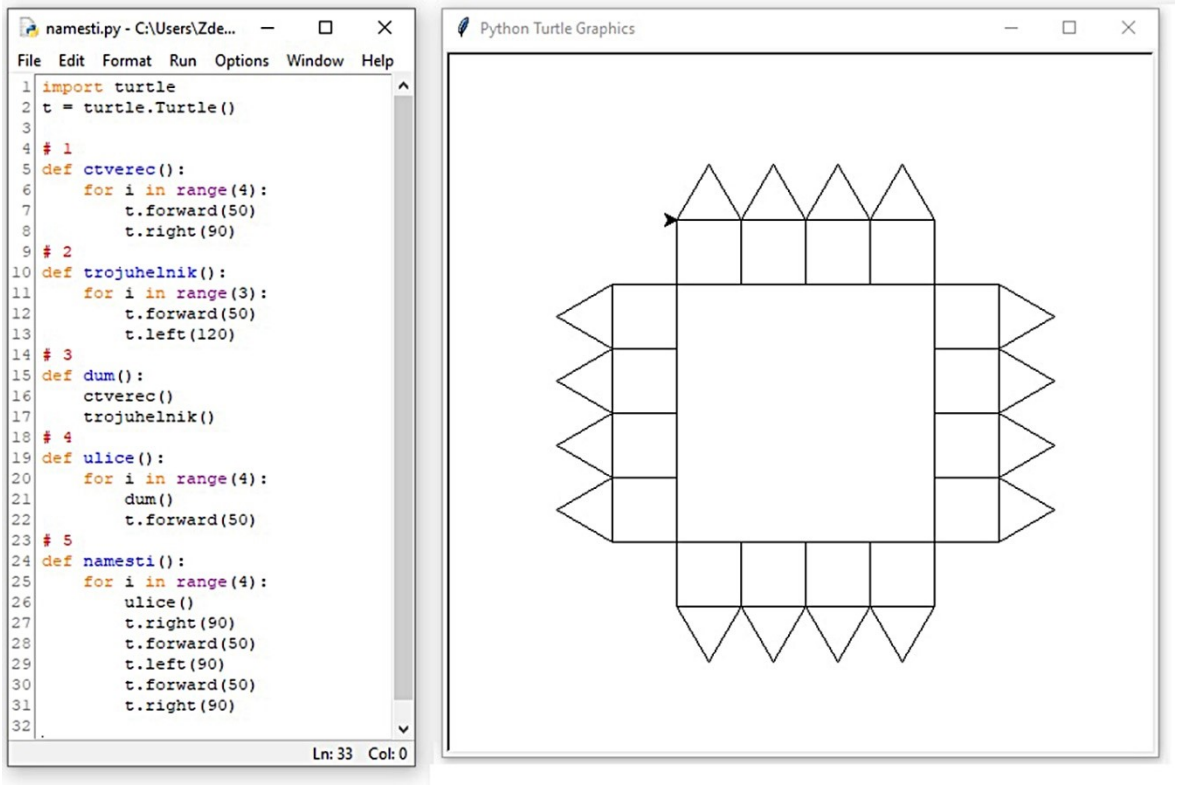
5.2 Dům, ulice, náměstí

V dalším kroku nakreslíme čtvercové náměstí se čtyřmi domy na každé jeho straně. Při této příležitosti naučíme žáky definovat vlastní funkce. Je to velmi snadné. „*Python pracuje s funkcemi podobně jako většina dalších jazyků, ale neodděluje hlavičkové soubory ... nebo sekce rozhraní/implementace ....*[33] *Pokud potřebujete nějakou funkci, prostě ji deklarujete ... Deklarace funkce začíná klíčovým slovem def. Následuje jméno funkce a v závorce pak*

[33] Funkce v jazyku Python nedefinují typ návratové hodnoty. Neurčují dokonce ani to, jestli vracejí hodnotu nebo ne. [55, s. 48]

argumenty. Více argumentů se odděluje čárkami.“ Definování funkce je zakončeno dvojtečkou. [55, s. 48]

Nejprve si vytvoříme (deklarujeme) funkci, která nakreslí čtverec, a funkci, která nakreslí trojúhelník. Následně definujeme funkci, která ze čtverce a trojúhelníku vytvoří dům. Funkce jednoduše zavolá jiné dvě dříve definované funkce - ‚*ctverec*‘ a ‚*trojuhelnik*‘.[34] Správné fungování jednotlivých funkcí lze průběžně (při jejich definování) ověřovat jejich zavoláním. Jako čtvrtý krok vytvoříme funkci, která vytvoří ulici ze čtyř domů. Zde již nestačí pouze čtyřikrát zavolat funkci ‚*dum*‘, ale je potřeba přesunout kreslící želvu o délku strany dopředu, aby se nakreslil nový dům a neobtahoval se stále dům původní – viz řádek 22. A konečně jako pátý krok definujeme funkci ‚*namesti*‘. Zde opět nestačí pouze čtyřikrát zavolat funkci ‚*ulice*’, ale po nakreslení každé ulice je potřeba přesunout kreslící želvu do správného budu a správně ji orientovat, aby funkce vykreslila náměstí, nikoliv neustále stejným směrem pokračující ulici. [54, s. 18] Řešení ukazuje Obrázek 20.



Obrázek 20. Funkce ‚*namesti*’ sestavená z jiných funkcí

[34] Tento postup odpovídá postupu zdola nahoru, viz kapitola 3.5 a Tabulka 2. Existuje i možnost zapsat do zdrojového kódu volání doposud nedefinované funkce a postupovat shora dolů.

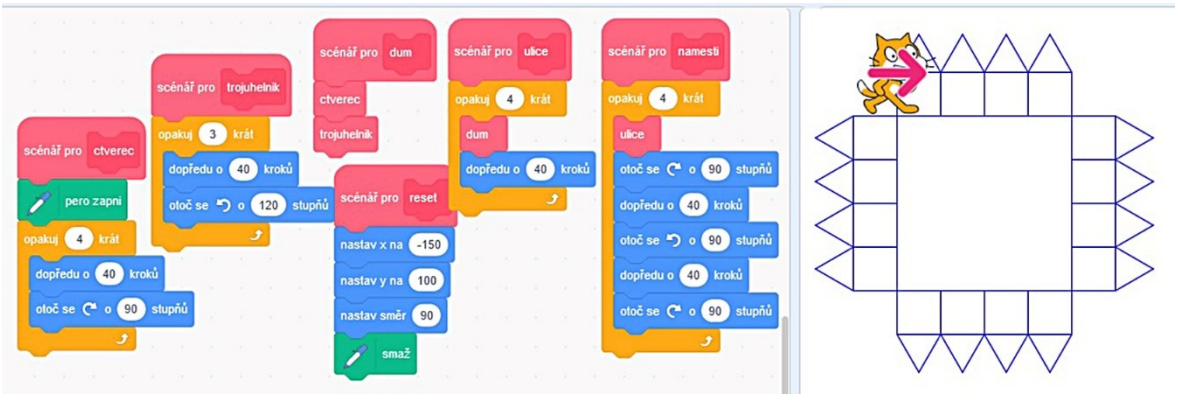
Program by bylo možné dále vylepšovat např. příkazy pro zvednutí a položení pera - *penup*, *pendown* – v momentech, kdy se kreslící želva přesunuje na novou výchozí pozici. „ … *želva za sebou nechává čáru, jen pokud má položené pero*.“ [52, s. 45] V našem případě jsou to příkazy na řádku 22 a na řádcích 28-30. Je možné uvažovat i rozšiřující příkazy, které například mění barvu pera nebo tloušťku čáry, ale pokud jde o učení a procvičování algoritmizace a psaní zdrojových kódů (programování) v Pythonu, není pro úplné začátečníky z nástrojů želví grafiky potřeba nic víc, než výše popsané základní příkazy.

K příkladu se lze časem vrátit a vylepšit jej o možnost zadání počtu domů v ulici při volání funkce ‚*namesti*', příp. zadat rozměr domu, nebo tento rozměr domu odvozovat od počtu domů v ulici (čím více domů, tím menší), aby obrázek v případě většího počtu domů nebyl moc velký.

5.3 Scratch

Nástroj pro kreslení (*Pero*) má i nejrozšířenější programovací jazyk pro děti Scratch. Oproti Pythonu má práce ve Scratchi několik nevýhod:

- v základním nastavení sice vidíme, kde je momentálně umístěn kurzor (kočka), ale nevíme, kam je nasměrován, což dosti komplikuje práci; je sice možné pracovat se šipkou, která směr ukazuje, ale tu je potřeba přidat
- oproti Pythonu, v němž je resetování při zkoušení a „ladění“ programu jednoduché, ve Scratchi je potřeba, nebo přinejmenším vhodné, vytvořit si resetovací blok, tj. funkci, která vše smaže a vrátí kreslící nástroj vždy do původní pozice a polohy
- plátno (canvas) pro malování je poměrně malé, pokud pero při kreslení vyjede z vymezeného prostoru, obrázek je deformovaný



Obrázek 21. Blok ‚*namesti*' ve Scatchi [scratch.mit.edu a vlastní]

5.4 Fraktály

Opakování stejných kroků při kreslení je typické pro tzv. fraktály, kdy „… *za využití rekurze* [lze vykreslovat zajímavé obrázky] *Úlohy lze vždy řešit krátkým programem, který však může být hodně náročný na vymyšlení. … Fraktály jsou sobě-podobné útvary, tj. skládají se z dílčích částí, které jsou podobné fraktálu jako celku. Fraktály mají tedy rekurzivní charakter, a proto je přirozené vykreslovat je pomocí rekurze. … Z pohledu matematika je fraktálem útvar, který dostaneme pro n jdoucí do nekonečna. Pro vykreslování na počítači však pochopitelně potřebujeme konečné n, které je vstupem našich programů. … fraktály se* [ovšem] *nezabývají jen matematici*.“ [52, s. 48]

6 ŘEŠENÍ PROBLÉMŮ

„*Když rozumíte tomu, co děláte, nic se neučíte*.“[35] [52, s. 63] Lidstvo se snaží nalézt efektivní „*metody učení a vyučování ... od počátku své existence, neboť s rozvojem řeči můžeme hovořit rovněž o rozvoji záměrné výchovy a o rozvoji vzdělávání. Otázka vyučovacích metod patří mezi základní didaktické kategorie ...*“ [56, s. 8] Jedním z aktuálních trendů v pedagogice je učení prostřednictvím *řešení problémů* (problem solving). Součástí RVP, který nově mezi klíčové kompetence v etapě základního vzdělávání zavádí digitální kompetence, jsou zahrnuty rovněž kompetence k učení a kompetence k řešení problémů.[36] [3, s. 10]

6.1 Co je problém

Americký didaktik Jeremy Kilpatrick charakterizuje problém jako „ … *situaci, v níž máme dosáhnout nějaký cíl, ale přímá cesta k němu je blokována*.“ [57, s. 15] Pojem problém lze vysvětlit i jinak. „*Při tomto vysvětlení se budeme opírat o tři hlavní složky, které problém má. Jsou to:*

1. *Výchozí situace, v níž popisujeme souvislosti a poskytujeme informace nebo údaje.*
2. *Cíl, kterého chce řešitel dosáhnout.*
3. *Cesta od výchozí situace k cíli, která pro řešitele může (ale také nemusí) být zřejmá či dosažitelná.*

Pomocí těchto tří složek můžeme problémy rozdělit do několika skupin.“ [57, s. 15] Viz následující Tabulka 3.

[35] Výrok je připisován v podkapitole 2.5 zmiňovanému Abrahamu Lincolnovi: „*If you understand what you're doing, you're not learning anything*.“

[36] Dalšími klíčovými kompetencemi jsou, vedle kompetencí digitálních, kompetence komunikativní, kompetence sociální a personální, kompetence občanské a kompetence pracovní.

Tabulka 3. Dělení problémů [vlastní zpracování dle 57, s. 15-16]

Výchozí situace	Cíl	Cesta	Označení v češtině	Označení v angličtině
přesně popsána	přesně zadán	je známa	cvičení	routine problem
přesně popsána	přesně zadán	není známa	úloha	non-routine problem
přesně popsána	není přesně zadán nebo není zadán vůbec	není známa	zkoumání	research

„*Hranice mezi rutinními a nerutinními problémy není ostrá. V určitou chvíli může být pro některého studenta daný problém rutinní, pro jiného nerutinní. Navíc i u určitého jedince se může z nerutinního problému za určitých okolností stát problém rutinní (naučí se např. řešit určitý typ algebraických rovnic)*.“ [57, s. 16]

6.2 Pedagogická poznámka

Americký filozof, pedagog, psycholog a reformátor vzdělávání John Dewey (1859-1952), „*… kladl velký důraz na nabývání zkušeností z praktické činnosti*[37] *a na aktivní experimentování. Po získání zkušeností následuje poskytnutí či vlastní vyvození teoretického vysvětlení. Zkušenost tedy stojí v centru celého učebního procesu.* [58, s. 159] Jeho podporu metody vyučování prostřednictvím řešení konkrétních (praktických) úloh již jen podtrhuje výrok: „*Přemýšlíme jen tehdy, když čelíme problémům*.“

Pokud žák řeší nějakou úlohu samostatně a nedaří se mu najít správné řešení „*a ocitne se ve slepé uličce*“ [1, s. 115], vyvstává otázka, kdy, vzhledem k tomu, že samostatně objevené řešení problému je cennější než řešení zprostředkované, má pedagog zasáhnout, tj. napovědět, poradit či dokonce prozradit správné řešení, aby frustrovaného žáka neodradil od zájmu o daný předmět.

„*Jaký typ intervence zvolit? Na jaké rovině je třeba podávat vysvětlení? Kam až je třeba se „vrátit", aby byla pomoc účinná*?“ [1, s. 115] Učitel by měl odolat pokušení poradit žákovi či dokonce ukázat mu řešení příliš brzy. [59, s. 6] Z pedagogického hlediska je totiž nejcennější, když si žák přijde (byť s pomocí učitele) na správné řešení sám.[38] „*Bohužel*

[37] právě s Deweyem je spojováno heslo *Learning by doing* - učení se prostřednictvím (praktické) činnosti
[38] Konec konců, heslem Montessori pedagogiky je: „*Pomoz mi, abych to dokázal sám*.“

většina lidí má jen velmi malé povědomí o tom, jak analyzovat problém a jaké strategie lze použít k jeho řešení.“ [60] Pokud učitel sezná, že je opravdu načase žákovi napovědět, je potřeba, aby věděl jak, protože mnohdy „*… ani ti, kteří se stali odbornými řešiteli, nejsou v mnoha případech schopni říci, co je vedlo k tomu, že zvolili jedno řešení místo druhého*.“ [60]

V současnosti se v pedagogice, a to nejen v zahraničí, ale i u nás, objevuje trend nedávat žákům domácí úkoly. [61] Tento trend, “nezatěžovat” děti školou a školními povinnostmi mimo vyučování, tj. aby dítě, pokud zrovna není ve škole, na školu aktivně nemyslelo, si neodporuje s jiným aktuálním psychodidaktickým trendem, který považuje za důležitou součást procesu učení spánek. [56, s. 33]

„*Důležitost spánku pro proces učení je implicitně známa již dlouhou dobu*“ [56, s. 33], proto již ve starší literatuře najdeme důležitý postřeh, že zcela zásadní podmínkou pro „pasivní“ vyřešení problému, tj. že člověka napadne řešení nějakého problému v okamžiku, kdy na něj (aktivně) nemyslí, třeba i ve spánku (ve snu), je schopnost „*zpaměti přesně reprodukovat zadání příkladu*“ nebo zkrátka problém podrobně zpaměti popsat. [62, s. 124]

Úlohy na webových stránkách umimeinformatiku.cz a neobsahují řešení.[39] Žák by se měl pokusit najít řešení samostatně a nenechat se odradit „*po prvním soustředěném úsilí příklad řešit*“, zkusit „*... jinou úlohu a pak se později … k té nedořešené*“ vrátit. [59, s. 6; 62, s. 124] „*Každý - třeba i neúspěšný - vlastní pokus … dá mnohem víc než přečtení hotového řešení*.“ [24, s. 12] Některé komentáře k vybraným úlohám v praktické části obsahují tipy, jak konkrétně žákovi, který si neví rady, napovědět.

[39] Důvodem může být i to, že možných řešení je více. „*Dobrý programátor musí ‚vidět' pro každou úlohu několik dobrých řešení a být schopen správně odhadnout potíže, se kterými se v každém z nich může setkat. Proto vám většinou přinese vyřešení jedné úlohy několika (skutečně rozdílnými) způsoby a následná diskuse o nich daleko více, než vyřešení více úloh prvním způsobem, který vás napadl*.“ [24, s. 12-13]

II. PRAKTICKÁ ČÁST

7 BINÁRNÍ ČÍSLA

„*Od nepaměti počítáme v desítkové soustavě. Počítání ‚po deseti' s použitím poziční hodnoty každé číslice se nám zdá být nejjednodušší ze všech způsobů počítání. Jsme ochotni tvrdit, že všechny národy ve všech dobách počítali tímto způsobem. V Babylónii … se počítalo pravděpodobně na tucty, přičemž základní jednotkou bylo pět tuctů čili 60.*[40] *… Desítková soustava má řadu předností jak při písemném, tak i ústním výpočtu. Avšak teoreticky má dvanáctková soustava více výhod. Deset je dělitelné 2 a 5, zatímco 12 je dělitelné 2, 3, 4 a 6*." [42, s. 100]

„*… zatímco my jsme zvyklí počítat a zobrazovat čísla v desítkové soustavě, počítač používá soustavu o základu 2 (tzv. dvojkovou nebo také binární soustavu).*" [50, s. 37] Počítače jsou elektronické stroje, v jejichž elektronických obvodech mohou být jen dva stavy – buď jimi prochází, nebo neprochází proud. Je pro ně tudíž velmi výhodné využití dvojkové soustavy, která pracuje jen se dvěma číslicemi, s nulou a jedničkou. Dvojkovu soustavu vynalezl (již) v 17. století německý filosof a matematik Gottfried Wilhelm Leibniz (1646-1716) [63, s. 185] a zřejmě tušil, že bude vhodná pro použití ve výpočetní technice,[41] neboť tvrdil, že „*… není hodné vynikajících mužů, aby trávili čas počítáním jako otroci. To je činnost, která by beze všeho mohla být přenechána komukoliv jinému, pokud by byly použity stroje*."[42] [64, s. 5]

7.1 Dvojková soustava

Pro pochopení dvojkové soustavy navrhují autoři metody CSunplugged, která sice pracuje bez počítačů, ovšem plně využívá moderní tzv. aktivizační výukové metody, vyrobit jednoduché papírové karty s jednou, dvěma, čtyřmi, osmi, šestnácti (tj. vždy dvojnásobkem předchozího počtu) tečkami na jedné straně. Karty následně poskládáme zleva doprava od

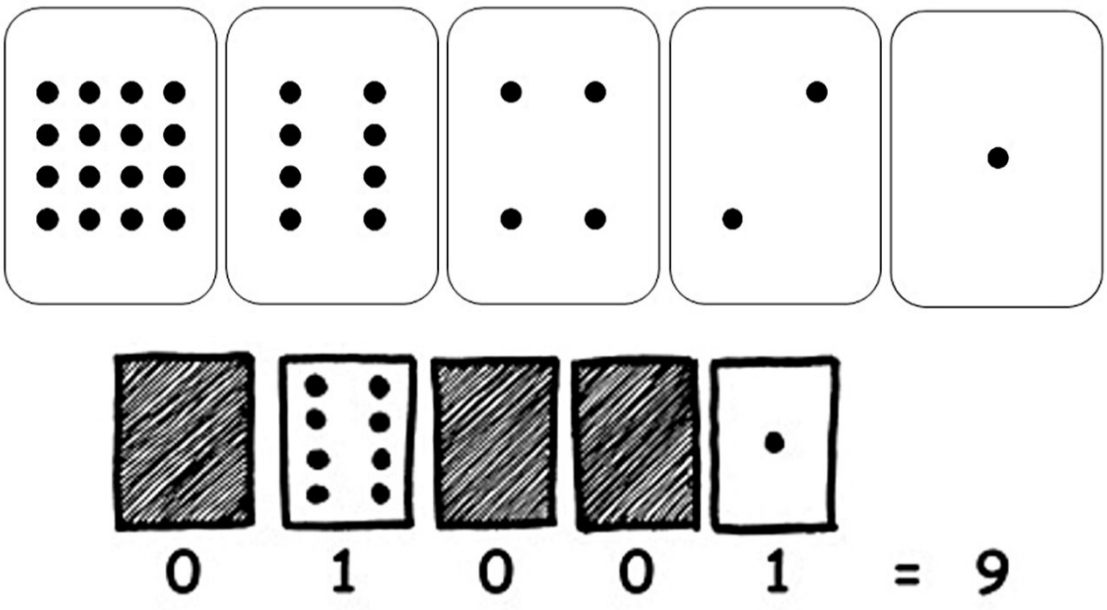
---

[40] Např. Velká Británie přešla na decimální systém dělení své měny, libry šterlinků (£), až v roce 1971. Do té doby se jedna libra dělila na 20 šilinků a jeden šilink na 12 pencí. (Jedna libra šterlinků tak sestávala z 240 pencí.) U britských délkových jednotek se jedna míle rovná 1760 yardům, jeden yard se skládá ze 3 stop a jedna stopa odpovídá 12 palcům.

[41] Vedle algoritmů „starých Řeků" další příklad, kdy teoretický objev předběhl dobu a praktické využití se pro něj našlo až později.

[42] „*It is unworthy of excellent men to lose hours like slaves in the labor of calculation which could safely be relegated to anyone else if machines were used*."
Většina předchůdců dnešních počítačů vznikla právě proto, aby ulehčila provádění rutinních (numerických) výpočtů – Konrad Zuse, William Seward Burroughs, Leon Bollee … [64, s. 21]

karty s nejvyšším počtem teček po kartu s nejnižším počtem teček.[43] Karta obrácená stranou bez teček směrem nahoru představuje nulu, karta obrácená stranou s tečkami směrem nahoru představuje nulu, viz Obrázek 22. [23, s. 4]



Obrázek 22. Binární karty [23, s. 5 a 7]

Následně je žákům uloženo, aby vyjádřili počtem zobrazených teček, tj. obracením karet, různá čísla. Žák se tak přesvědčí, že tímto způsobem lze vyjádřit jakoukoliv číselnou hodnotu v daném rozsahu a zároveň, že každá hodnota může být vyjádřena pouze jedním jediným způsobem. [23, s. 5 a 15] V poslední fázi aktivního objevování binárního systému začne žák zapisovat zadané hodnoty v desítkové soustavě v soustavě dvojkové.

7.2 Dvě zajímavosti

Pokud žáci základní princip fungování dvojkové soustavy pochopili, lze je (třeba na závěr hodiny) upozornit na dvě zajímavosti v souvislosti s binárními čísly. U první poznámky budeme vycházet ze znalostí žáků desítkové soustavy. Jestliže přidáme vpravo za číslo v desítkové soustavě nulu, dostaneme desetinásobek původní hodnoty. Např. když k 9 připíšeme nulu, dostaneme 90, z 30 dostaneme 300. Pokud dopíšeme nulu vpravo za

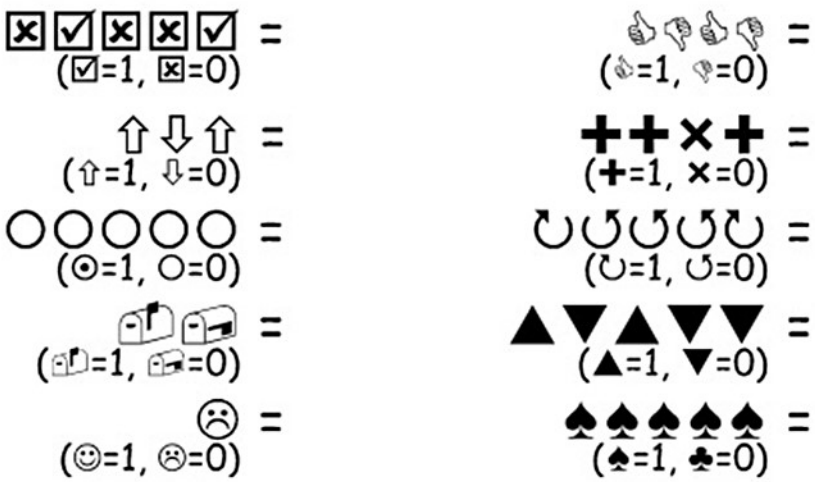
[43] Dvojková soustava je totiž, stejně jako desítková, tzv. poziční, záleží tedy na pozici, na které se daný znak nachází.

číslo ve dvojkové soustavě, (analogicky) získáme dvojnásobek původní hodnoty, např. $1001_{(2)}$ je v desítkové soustavě 9 a $10010_{(2)}$ je v desítkové soustavě 18.[44] [23, s. 12]

Druhá zajímavost tak trochu navazuje na předchozí, a sice, ačkoliv ne vždy poznáme na první pohled na číslo zapsané ve dvojkové soustavě, jakou hodnotu představuje, bezpečně můžeme okamžitě určit, zda je dané číslo sudé nebo liché. U sudých čísel zapsaných ve dvojkové soustavě je vždy první číslo zprava nula, zatímco u lichých čísel je vždy prvním číslem zprava jednička.[45] Pokud si dáme oba výše uvedené poznatky dohromady, potvrzují známý fakt, že vynásobíme-li liché číslo dvěma, dostaneme číslo sudé.

7.2.1 Cvičení

Žák zapíše zakódovaná binární čísla, viz Obrázek 23, nejprve ve dvojkové soustavě a potom hodnoty převede do soustavy desítkové.



Obrázek 23. Zakódovaná binární čísla [23, s. 12]

[44] Pokud bychom k původní hodnotě připsali dvě nuly (100100), vznikl by čtyřnásobek původního čísla (36).
[45] Z tohoto pohledu se první bit zprava může jevit jako velmi důležitý, protože existuje mnoho algoritmů, např. z oblasti teorie čísel, jejichž součástí je rozlišování mezi sudým a lichým číslem. Teorie výpočetní techniky ovšem tento bit označuje za nejméně významný bit - *least significant bit* (LSb).

7.2.2 Řešení

$01001_{(2)} = 9$	$1010_{(2)} = 10$
$101_{(2)} = 5$	$1101_{(2)} = 13$
$0000 = 0$	$10001_{(2)} = 17$
$10_{(2)} = 2$	$10100_{(2)} = 20$
$0_{(2)} = 0$	$11111_{(2)} = 31$

7.2.3 Poznámky z praktické výuky

Požadavek převodu binárního kódu odvozeného z obrázku do desítkové soustavy v literatuře, z níž je čerpáno [23], není. Žáci 4. ročníku, kterým byly při praktickém testování v konstruktivistické části vyučovací hodiny rozdány pracovní listy, a na tabuli bylo ponecháno grafické zobrazení binárních karet, viz Obrázek 22, z úvodní, transmisivní části vyučovací hodiny, úlohu, byť byla oproti předloze mírně ztížená, zvládli.

Problém činil pouze hned první příklad, kdy číslo zapsané ve dvojkové soustavě začíná nulou. Vzhledem k tomu, že žákům bylo vysvětleno, že při převodu z binární do desítkové soustavy se postupuje zprava doleva, což je potřeba zdůraznit, měli žáci pocit, že číslo se také tak čte, protože na to, aby číslo začínalo nulou, nejsou z desítkové soustavy zvyklí. Pro modifikaci s doplněním úlohy o převod čísla z dvojkové do desítkové soustavy (požadavek, který v původním zadání nebyl) bude možná lepší první symbol zleva (pro nulu) smazat, aby daný kód, stejně jako všechny zbývající, začínal jedničkou.[46]

Vzhledem k tomu, že látka je předkládána žákům ve 4. ročníku[47], je ve výuce vhodnější používat srozumitelnější terminologii *desítková* a *dvojková soustava*, než *decimální* a *binární systém*.

[46] Tak či tak nejsou všechny kódy v daném cvičení v pětimístném (pětibitovém) formátu.
[47] Autoři koncepce dokonce tvrdí, že konkrétně tato látka je vhodná pro ještě mladší děti. [23, s. 3]

8 ŘAZENÍ A TŘÍDĚNÍ

„*…řazení je zcela zásadní pro práci s téměř jakýmikoli informacemi. Ať už se jedná o hledání nejvyšší či nejnižší hodnoty, nejběžnější nebo nejvzácnější hodnoty, sčítání, indexování, označování duplicit nebo jednoduše pouhé vyhledávání požadované položky, vše obecně začíná řazením*.“ [32, s. 72] „*Řazení je jedním ze základních stavebních kamenů chodu počítače. Ve skutečnosti právě řazení v mnoha ohledech přivedlo počítače k životu*.“ [32, s. 71]

„*Google a Bing označujeme za vyhledávače, ale takové označení je spíše nevhodné - ve skutečnosti se jedná o řadiče. Google nezískal prvenství v poskytování přístupu k informacím z celého světa ani tak díky tomu, že náš text dokáže najít na stovkách milionů webových stránek (totéž obecně dokázali dost dobře i jeho konkurenti v 90. letech), ale proto, že tak dobře řadí nalezené stránky a ukazuje nám pouze nejrelevantnějších deset výsledků*.“ [32, s. 72]

„*Algoritmům se říká třídící, protože se prvky setřídí (seřadí) podle velikosti. Slovo sort v angličtině znamená právě setřídit nebo seřadit; uspořádat. … Třídících algoritmů je celá řada*.[48] *… Každý třídící algoritmus nakonec vrátí setříděnou množinu prvků …, ale každý z nich na to jde jinak. Každý využívá jinou myšlenku*.“ [16, s. 41] Nejjednodušším na pochopení fungování i na pochopení fungování algoritmu jako takového je BubbleSort.[49] Bohužel ale patří mezi nejméně efektivní. [16, s. 41] „*Jiné třídící algoritmy jsou … daleko efektivnější, jejich algoritmy jsou však mnohem méně průhledné*.“ [16, s. 41] „*Nejjednodušší třídicí algoritmy patří do skupiny tzv. přímých metod. Všechny mají několik společných rysů: Jsou krátké, jednoduché a třídí na místě*.“ [33, s. 61] Mezi ty nejdůležitější patří: QuickSort, SelectSort, InsertSort a MergeSort. [16, s. 41]

8.1 BubbleSort

Třídící algoritmus BubbleSort, jehož „*… základem je myšlenka nechat stoupat větší prvky v poli podobně, jako stoupají bublinky v limonádě*“ [33, s. 62], prochází několikrát celý

[48] Striktně vzato, termín řazení je správnější. Data totiž nerozdělujeme do nějakých tříd, nýbrž je uspořádáváme, tedy řadíme, dle určitého kritéria. Ovšem pojem třídění je mezi českými informatiky natolik zažitý, že je bláhové chtít na tom cokoliv měnit. [33, s. 61]

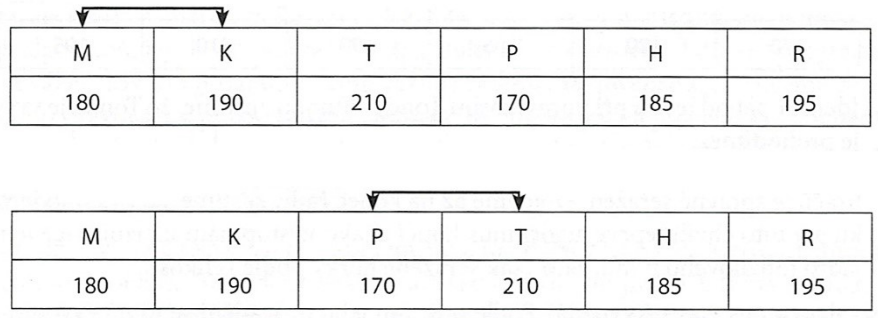
[49] Třídící algoritmy jsou zároveň dobrým konkrétním příkladem pro výklad složitosti algoritmů.

seznam tříděných prvků a porovnává sousední hodnoty [23, s. 77], zda je prvek vlevo nižší než jeho „*…pravý soused (kromě toho úplně vpravo, jelikož ten už nemá vpravo souseda) … Pokud je* … [prvek] *vlevo vyšší než jeho soused vpravo, prohodí je*.“ [16, s. 41]

Prakticky si ukážeme na příkladu šestičlenného volejbalového týmu, jehož členy jsou:

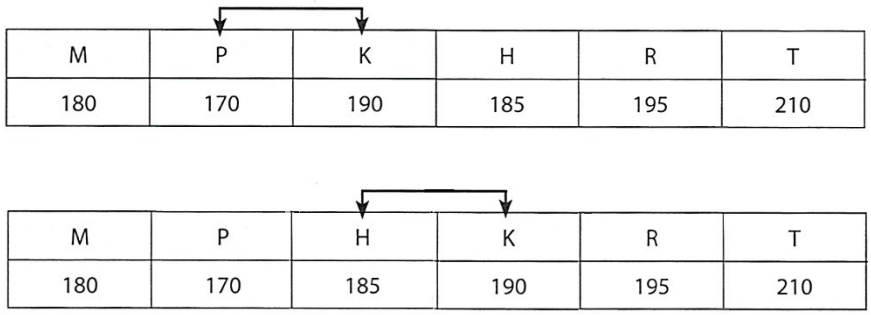
- Karel měří 190 cm
- Mirek měří 180 cm
- Tonda je nejvyšší, měří 210 cm
- Pepa je nejnižší, měří 170 cm
- Honza měří 185 cm
- Ruda měří 195 cm

Hráče si seřadíme podle velikosti tak, aby vlevo byl nejnižší a vpravo nejvyšší. Nejprve porovnáme Karla s Mirkem. Karel je vyšší, proto se prohodili. Tonda je vyšší než Karel, proto zde k prohození nedošlo. Protože je ale Pepa menší než (nejvyšší) Tonda, došlo, jak ukazuje Obrázek 24, k prohození Pepy a Tondy na 3. a 4. pozici. [16, s. 41]



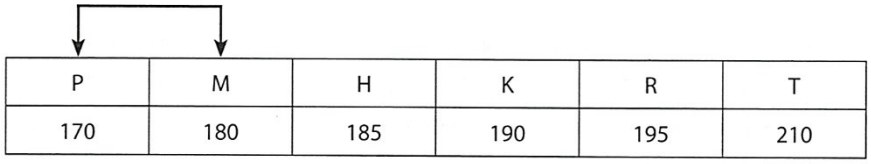
Obrázek 24. Průběh třídícího algoritmu BubbleSort - 1. průchod [16, s. 39]

Následně se prohodili Tonda s Honzou a, protože Tonda je nejvyšší, je tudíž vyšší než Ruda, prohodil se i s ním. Na konci 1. průchodu tedy máme stav M, K, P, H, R a T. Při 2. průchodu vidíme, že dvojice M a K je v pořádku, takže zde k prohození nedojde, ale již na 2. a 3. pozici k prohození dojde, protože Karel je vyšší než Pepa. V dalším kroku 2. průchodu se prohodí Karel a Honza – viz Obrázek 25.



Obrázek 25. Průběh třídícího algoritmu BubbleSort - 2. průchod [16, s. 40]

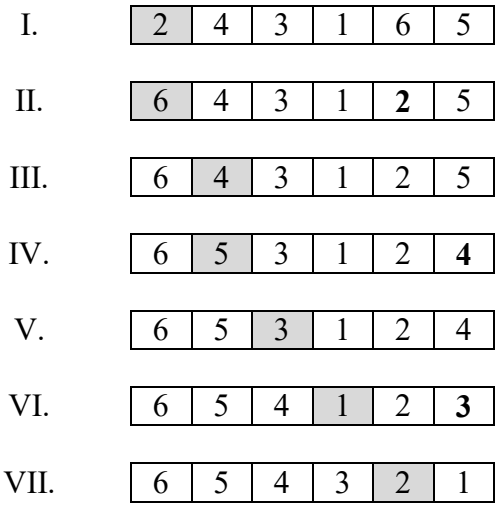
Při 3. průchodu se prohodí Pepa a Mirek a 1. a 2. pozici, dále, po zbytek 3. průchodu již k dalším prohozením nedojde – viz Obrázek 26. Nyní, po třech průchodech, máme hráče seřazené od nejmenšího po nejvyšší, ale „… *algoritmus tím nekončí. Protože jsme právě prohodili dva hráče mezi sebou, musíme pěkně od leva až doprava* [projít celou řadu ještě jednou a] *zkontrolovat, že každý pár sousedních hráčů je správně seřazen*.“ [16, s. 40] Algoritmus (a tedy běh programu) končí až poté, co během celého průchodu nebyla provedena žádná změna.



Obrázek 26. Průběh třídícího algoritmu BubbleSort - 3. průchod [16, s. 40]

8.2 SelectSort

„*SelectSort je snad ještě jednodušší než Bubblesort. Je založen na výběru nejvyššího [nebo nejnižšího] prvku. Postavíme se na začátek nesetříděného pole, úplně doleva. Šedivě je naznačeno místo, kde stojíme*.“ [16, s. 153]



Obrázek 27. Průběh třídění podle algoritmu SelectSort [16, s. 154]

I. a II.

„*Hledáme nejvyšší prvek celého pole. Až ho najdeme, vložíme ho na místo, kde stojíme. Tedy na první místo, úplně doleva. Prvek, který tu stál původně, se s nejvyšším prostě prohodí. Původní prvek, který mění svou pozici s nejvyšším prvkem, označujeme v poli tučně*.“ [16, s. 154]

III. a IV.

„*Posuneme se o jeden prvek dál (na druhé místo). A opět hledáme nejvyšší prvek ze zbytku pole (od místa, kde stojíme směrem doprava). Až najdeme nejvyšší prvek zbytku pole, vložíme ho na místo, kde stojíme*.“ [16, s. 154]

V., VI. a VII.

„*Potom se posuneme o jedno místo doprava. Tyto kroky provádíme tak dlouho, dokud nedojdeme na předposlední místo pole. Poslední prvek můžeme … vynechat, protože už se nemá s čím porovnávat – je úplně vpravo a nemá ani, s kým by se prohodil*.“ [16, s. 154]

8.3 QuickSort

„*QuickSort je ještě mnohem rychlejší než SelectSort, zvláště u delších seznamů. Vlastně je to jedna z nejlepších známých metod.*“[50] [23, s. 75] Toto algoritmické řešení odpovídá mezi

[50] „*Quicksort is a lot faster than selection sort, particularly for larger lists. In fact, it is one of the best methods known*.“

programátory oblíbenému, a tudíž často používanému, přístupu zvanému „rozděl a panuj“ (*divide and conquer*).[51] [52, s. 18]

Princip jeho fungování si ukážeme na příkladu seřazení závaží od nejlehčího po nejtěžší. „*Vyberte si náhodně jedno závaží a umístěte ho stranou. Nyní s ním porovnejte všechna zbývající závaží. Všechna lehčí dejte doleva, doprostřed zvolené závaží a těžší závaží dejte doprava. (Může se stát, že na jedné straně bude výrazně více závaží než na straně druhé.)*[52] [23, s. 75]

Vyberte jednu ze skupin a celý postup opakujte. Totéž udělejte i s druhou skupinou. Nezapomeňte, že jedno závaží musí být vždy uprostřed. Opakujte tento postup u zbývajících skupin, dokud nebude každá skupina víc než jedno závaží. V okamžiku, kdy budou všechny skupiny rozděleny na jednotlivá závaží, budou tato závaží roztříděna od nejlehčího po nejtěžší.“[53] [23, s. 75]

8.4 InsertSort

„*InsertSort funguje tak, že přesune každý objekt z nesetříděné skupiny na správné místo v neustále se zvětšující skupině (viz obrázek níže). Každým vložením se skupina nesetříděných objektů zmenšuje a skupina setříděných zvětšuje, dokud není vše setříděno. Tuto metodu často používají karetní hráči, aby si uspořádali karty v ruce.*“[54] [23, s. 77]

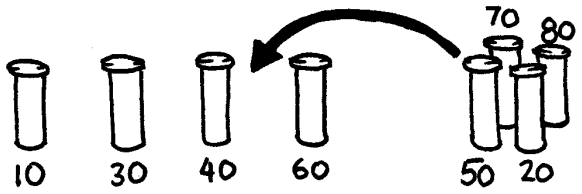
---

[51] z latinského *divide et impera* – tímto heslem se řídili již vládci ve starověkém Římě

[52] „*Choose one of the objects at random, and place it on one side of the balance scales. Now compare each of the remaining objects with it. Put those that are lighter on the left, the chosen object in the middle, and the heavier ones on the right. (By chance you may end up with many more objects on one side than on the other.)*“

[53] “*Choose one of the groups and repeat this procedure. Do the same for the other group. Remember to keep the one you know in the centre. Keep repeating this procedure on the remaining groups until no group has more than one object in it. Once all the groups have been divided down to single objects, the objects will be in order from lightest to heaviest.*”

[54] „*Insertion sort works by removing each object from an unsorted group and inserting it into its correct position in a growing list (see picture below). With each insertion the group of unsorted objects shrinks and the sorted list grows, until eventually the whole list is sorted. Card players often use this method to sort a hand into order*.”



Obrázek 28. InsertSort [23, s. 77]

8.5 MergeSort

MergeSort je další z metod, která při uspořádání prvků vychází ze zásady ‚rozděl a panuj'.[23, s. 77] "*MergeSort bohužel neumí třídit na místě: při slévání musí být zdrojové běhy uloženy jinde než cílový běh. Proto potřebujeme pomocnou paměť ..., například v podobě pomocného pole stejné velikosti jako vstupní pole*." [33, s. 65]

„*Nejprve je skupina náhodně rozdělena na dvě stejně veliké skupiny (případně přibližně stejné, pokud máme lichý počet prvků). Každá z těchto skupin je setříděna a tyto dvě skupiny jsou pak sloučeny od jedné. Sloučení dvou setříděných skupiny je pak snadné – vezmeme vždy prvek s nižší hodnotou na kraji dvou skupin*."[55] [23, s. 77]

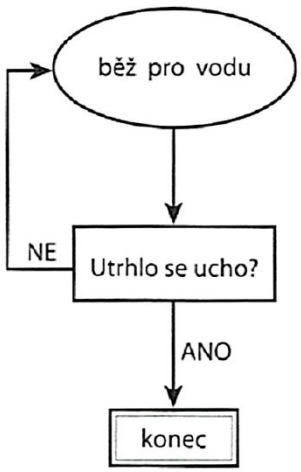
8.5.1 Cvičení

Tyto algoritmy si lze prakticky vyzkoušet s žáky, kteří se budou fyzicky řadit podle fyzické výšky, podle abecedy (křestního jména, příjmení) apod. Opět se jedná o tzv. aktivizační výukovou metodu, konkrétně názorně demonstrační, v níž je žák „… *aktivním činitelem celého procesu, převážně se učí samostatným objevováním a zjišťováním informací, učí se vyhledávat a zpracovávat informace, aktivně spolupracuje s ostatními žáky, učí se týmové spolupráci, organizaci, kooperaci a komunikaci s lidmi v týmu*." [65, s. 55]

Jak již bylo řečeno výše, třídící algoritmy jsou vhodným materiálem pro pochopení základů algoritmizace. Zároveň jsou jedním ze způsobů vyjádření algoritmu, vedle např. slovního popisu, vývojové diagramy – viz podkapitola 4.2.3. Žák by tak měl být schopný rozpoznat, které přísloví je vyjádřeno pomocí následujícího vývojového diagramu – viz Obrázek 29.

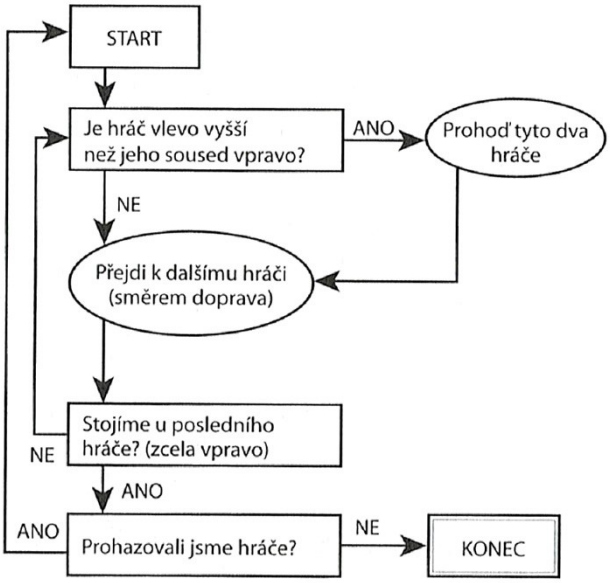
---

[55] „*First, the list is divided at random into two lists of equal size (or nearly equal if there are an odd number of items). Each of the two half-size lists is sorted, and the two lists are merged together. Merging two sorted lists is easy—you repeatedly remove the smaller of the two items at the front of the two lists*."

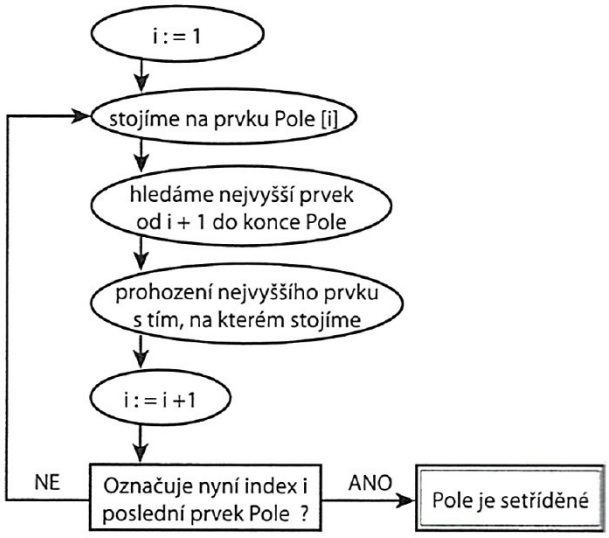


Obrázek 29. České přísloví vyjádřeno vývojovým diagramem [16, s. 95]

Stejně tak by žák měl poznat, které z výše popsaných třídících algoritmů jsou vyjádřeny na následujících vývojových diagramech – viz Obrázek 30 a Obrázek 31. Poslední úloha je již poněkud obtížnější.



Obrázek 30. Vývojový diagram – třídící algoritmus BubbleSort [16, s. 41]



Obrázek 31. Vývojový diagram – třídící algoritmus SelectSort [16, s. 154]

8.5.2 Řešení

Diagram Obrázek 29 – Tak dlouho se chodí se džbánem pro vodu, až se ucho utrhne.

Diagram Obrázek 30 – BubbleSort

Diagram Obrázek 31 – SelectSort

8.5.3 Poznámky z praktické výuky

Při probírání jednotlivých třídících algoritmů je rozhodně lepší probírat je postupně jeden po druhém a prakticky procvičovat volbou jiných klíčů.[56] Při použití názorně demonstrační aktivizační metody a volbě třídícího klíče jako je fyzická výška žáka má učitel dohlížející na správnost průběhu vcelku snadnou úlohu. Při volbě dalších klíčů, jako např. jméno nebo příjmení žáka, je již důležité, aby učitel žáky dobře znal jmény. Metoda tudíž není vhodná pro praktikanty nebo pro úvodní dny na informatiku zaměřených letních táborů, kde nejen vedoucí děti ještě dobře neznají, ale kde se dokonce neznají ani děti navzájem.

Orientace v jednodušších vývojových diagramech po náležitém výkladu nečiní žákům na 2. stupni žádné větší potíže. Ostatně zkušenosti ze Spojených států amerických ukazují, že ne docela triviální vývojový diagram dokázala vytvořit osmiletá holčička, viz podkapitola

[56] Výklad a procvičování různých třídících algoritmů současně podle jednoho stejného klíče vnese žákům do učiva zmatek

4.2.3 této práce. Rozpoznání přísloví znázorněného pomocí vývojového diagramu je tak spíše otázkou znalosti onoho přísloví než nezvládnutí orientace ve vývojových diagramech.

U vývojových diagramů pro třídící algoritmy je potřeba žákům předem zopakovat jejich principy a upozornit, že hlavním klíčem k rozpoznání, o jaký algoritmus se jedná, je v prvním případě větvení u otázky, zda je hráč vlevo vyšší než jeho soused vpravo, a ve druhém případě pokyn k prohození nejvyššího prvku s tím, na kterém stojíme.

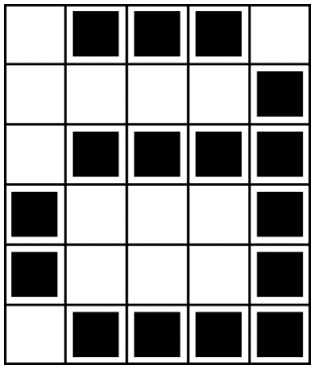
9 KÓDY A KOMPRESE DAT

Kromě samotných čísel lze zakódovat i písmena, obrázky nebo třeba zvuk. Obrázky a zvuk jsou rovněž dobrým příkladem pro vysvětlení základních principů komprese dat.

9.1 Mřížka

Budeme pracovat s mřížkou o rozměru 5 × 6 čtverečků. Počítačový monitor je rozdělen právě na taková malá políčka, kterým se říká pixely. Termín *pixel* vznikl, podobně jako termín *bit*, viz závěr podkapitoly 7.1, spojením anglických slov *pix*el *el*ement – prvek obrázku. Takto vytvořenému obrázku s pixely v mřížce se říká bitmapa, nebo také bitmapová či rastrová grafika. „*Obrázek je zde reprezentován mřížkou n × m bodů, velikost mřížky udává rozlišení*[57] *(kvalitu) obrázku. … Pokud obrázek v bitmapové grafice zvětšujeme, stává se zrnitým*.“ [52, s. 22]

Pokud bychom chtěli vyjádřit písmeno malé *a*, a pro binární kód by platilo pravidlo, že nula znamená černé pole a jednička pole bílé, pak by jednotlivé řádky zápisu kódy vypadaly následovně, viz Obrázek 32.

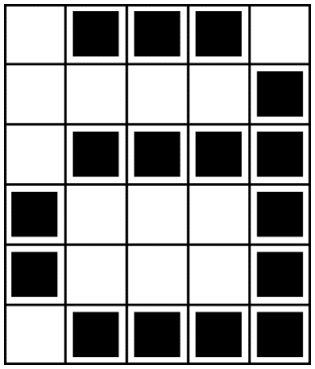


Obrázek 32. Mřížka s písmenem *a* bez komprimovaného kódu [23, s. 17]

Kód obsahuje 5 × 6, čili 30 znaků. Toto zakódování vyobrazení malého písmene *a* by ale bylo možné zkomprimovat, tj. zmenšit objem dat, téměř o polovinu, a sice na 17 znaků, i když tentokrát už si nevystačíme pouze s nulami a jedničkami. Pravidla pro komprimaci by mohla být třeba následující:

[57] rozlišení – angl. *definition*

První číslo na řádku udává vždy počet bílých políček. Pokud je první políčko, tedy pixel, černé, začíná kód daného řádku nulou, za níž následuje číselný údaj počtu černých pixelů. Každé v řádku následující číslo udává počet pixelů (políček) měnící se barvy. [23, s. 18] Viz Obrázek 33.



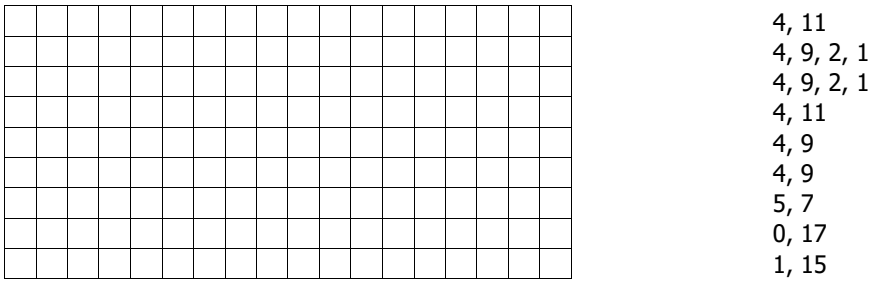
Obrázek 33. Mřížka s písmenem *a* s komprimovaným kódem [23, s. 17]

V podstatě jde o metodu „…*komprese s názvem RLE (run-length encoding, kódování podle délky). Jedná se o postup, kdy se posloupnosti stejných datových prvků nahrazují kódem, který udává opakovaný element a počet jeho následných opakování.* [Používá se v případech, kdy] *komprimovaná data obsahuji dlouhé sekvence stejných hodnot* …“ [2, s. 70]

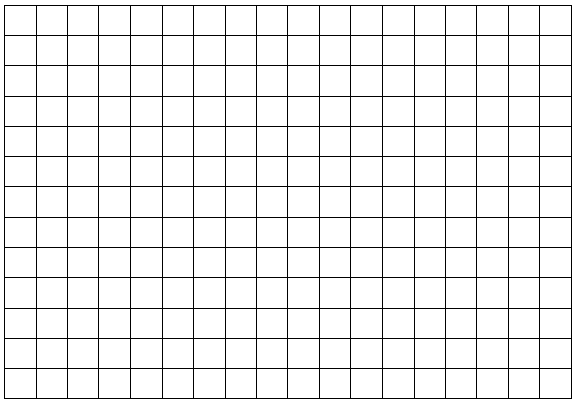
Komprimace, nebo také komprese, dat je výhodná nejen pro přenos, který je v případě menšího objemu dat rychlejší, ale také pro ukládání, protože menší datový soubor zabírá méně místa v paměti a je tudíž možné uložit více obrázků. [2, s. 69; 23, s. 24]

9.1.1 Cvičení

Následující cvičení obsahuje tři černobílé rastrové obrázky k doplnění podle číselných kódů. Úlohy jsou seřazené od nejlehčí po nejtěžší.

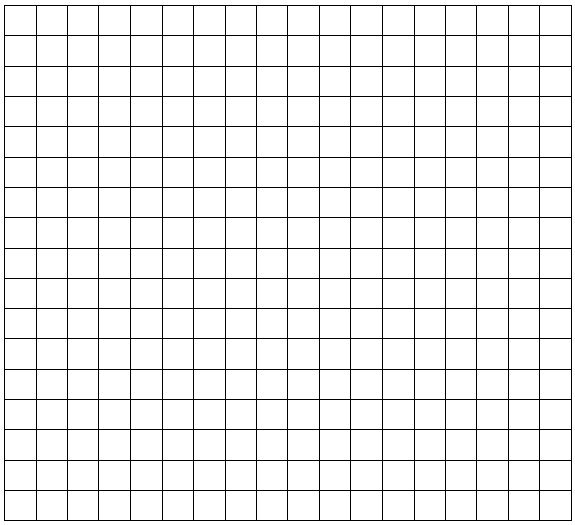


Obrázek 34. Zadání úlohy 9.1.1a [23, s. 20]



6, 5, 2, 3
4, 2, 5, 2, 3, 1
3, 1, 9, 1, 2, 1
3, 1, 9, 1, 1, 1
2, 1, 11, 1
2, 1, 10, 2
2, 1, 9, 1, 1, 1
2, 1, 8, 1, 2, 1
2, 1, 7, 1, 3, 1
1, 1, 1, 1, 4, 2, 3, 1
0, 1, 2, 1, 2, 2, 5, 1
0, 1, 3, 2, 5, 2
1, 3, 2, 5

Obrázek 35. Zadání úlohy 9.1.1b [23, s. 20]



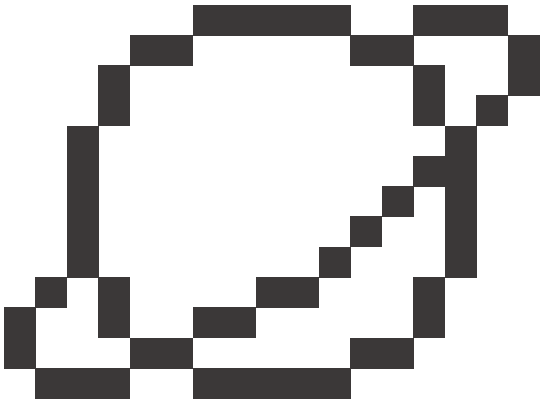
6, 2, 2, 2
5, 1, 2, 2, 2, 1
6, 6
4, 2, 6, 2
3, 1, 10, 1
2, 1, 12, 1
2, 1, 3, 1, 4, 1, 3, 1
1, 2, 12, 2
0, 1, 16, 1
0, 1, 6, 1, 2, 1, 6, 1
0, 1, 7, 2, 7, 1
1, 1, 14, 1
2, 1, 12, 1
2, 1, 5, 2, 5, 1
3, 1, 10, 1
4, 2, 6, 2
6, 6

Obrázek 36. Zadání úlohy 9.1.1c [23, s. 20]

9.1.2 Řešení

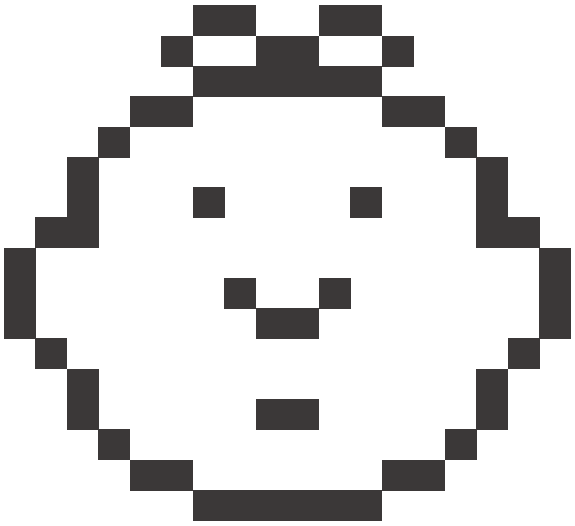
4, 11
4, 9, 2, 1
4, 9, 2, 1
4, 11
4, 9
4, 9
5, 7
0, 17
1, 15

Obrázek 37. Jednoduchý rastrový obrázek, úloha 9.1.1a [23, s. 25]



6, 5, 2, 3
4, 2, 5, 2, 3, 1
3, 1, 9, 1, 2, 1
3, 1, 9, 1, 1, 1
2, 1, 11, 1
2, 1, 10, 2
2, 1, 9, 1, 1, 1
2, 1, 8, 1, 2, 1
2, 1, 7, 1, 3, 1
1, 1, 1, 1, 4, 2, 3, 1
0, 1, 2, 1, 2, 2, 5, 1
0, 1, 3, 2, 5, 2
1, 3, 2, 5

Obrázek 38. Složitější rastrový obrázek, úloha 9.1.1b [23, s. 25]



6, 2, 2, 2
5, 1, 2, 2, 2, 1
6, 6
4, 2, 6, 2
3, 1, 10, 1
2, 1, 12, 1
2, 1, 3, 1, 4, 1, 3, 1
1, 2, 12, 2
0, 1, 16, *1*
0, 1, 6, 1, 2, 1, 6, *1*
0, 1, 7, 2, 7, *1*
1, 1, 14, 1
2, 1, 12, 1
2, 1, 5, 2, 5, 1
3, 1, 10, 1
4, 2, 6, 2
6, 6

Obrázek 39. Nejsložitější rastrový obrázek, úloha 9.1.1c [23, s. 25]

9.2 Poznámky z praktické výuky

Sami autoři upozorňují na obtížnost této úlohy a doporučují pouze lehké vybarvování tužkou a mít po ruce gumu. [23, s. 20] Úloha byla opět pro praktické testování ve 4. ročníku oproti předloze mírně modifikován, a sice (opět) mírně ztížena. Již při výkladu k mřížce s písmenem *a* s komprimovaným kódem, viz Obrázek 33, bylo upozorněno na to, že poslední číslo na každém řádku by mohlo být vynecháno.

Při vybarvování čtverečků (pixelů) by si tak žák měl všimnout, že, jako prvek komprese, je v zadání vynechán poslední číselný údaj, který doplňuje součet hodnot do počtu pixelů v řádku. (V tomto případě 18.) U prvního obrázku, Obrázek 37, je to na prvním řádku chybějící číslo 3, na druhém řádku číslo 2 atd. Tento údaj není pro správně vykreslení obrázku nutný. Tyto pixely zůstávají bílé.

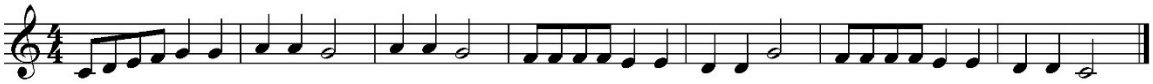
Spornější je případ jedniček na konci řádků 9-11 (kurzivou) u třetího obrázku, viz Obrázek 39. Tyto řádky končí černými pixely, jimž předchází pixely bílé. Také v tomto případě lze tyto jedničky považovat za zbytečné (zvláště u systémů, které znají a pracují s počtem pixelů v řádku), protože dává-li uvedený součet pixelů v daném řádku hodnotu nižší než celkový počet pixelů v řádku, je (v případě černobílých obrázků) zřejmé, že zbylé pixely do konce řádku jsou opačné barvy než poslední udaný pixel.[58] Navíc, bude-li žák sledovat vznikající obrázek, měl by si všimnout, že je symetrický podle vertikální osy a i to by mu mělo napovědět, že na konci oněch řádků něco chybí. Vedle dopočítávání pixelů na konci řádků činila problém i orientace v řádcích začínajících nulou, tj. černým pixelem.

Šikovnější žáci 4. ročníku úlohu zvládli během vyučovací hodiny, ti ostatní dostali dodělání za domácí úkol.

9.3 Modely, kódy, šifry

Na následujícím příkladu budou vysvětleny hned tři okruhy učební látky. Žáci se tak naučí různě nahlížet na stejný výukový materiál, pracovat s ním a postupně rozvíjet, rozšiřovat a aktualizovat své poznatky. [1, s. 108]

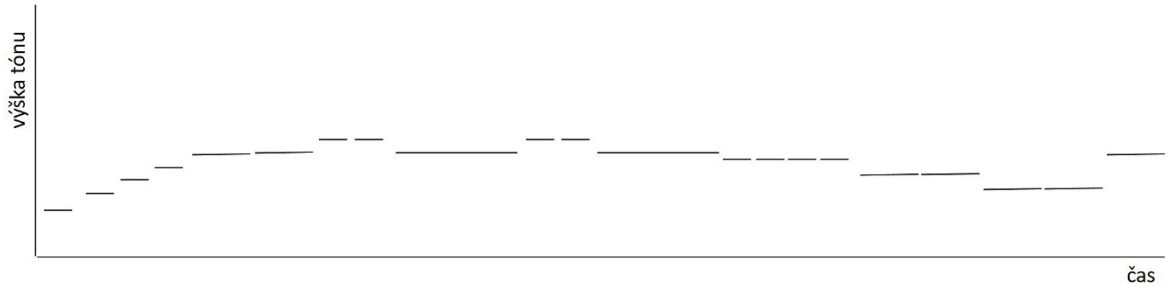
V učebnici *Základy informatiky pro 2. stupeň ZŠ* autorů Berki a Drábková [66] najdeme v kapitole *Modely* na straně 35 úloha vytvořit model notového zápisu písničky *Kočka leze dírou*. [66, s. 35]



Obrázek 40. Notový zápis písničky v C dur [66, s. 35]

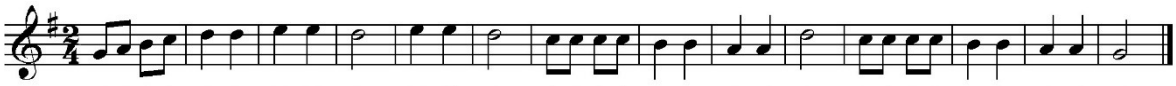
Jednou z možností je zápis do grafu, v němž vodorovná osa *x* zobrazuje čas, tj. začátek a délku trvání tónu) a svislá osa *y* zobrazuje výšku tónu. Tento způsob zápisu, přesněji řečeno grafického modelu, je dosti prostorově náročný, proto Obrázek 41 nezobrazuje melodii celou, pouze její část.

[58] Záležitost byla přes e-mailovou komunikaci konzultována s jedním z autorů skript CSunplugged, panem Timem Bellem.



Obrázek 41. Grafické znázornění melodie

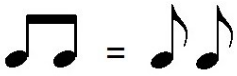
Zmiňovaná učebnice přináší i zápis stejné melodie v jiném taktu a jiné tónině. Výše uvedený obecný grafický model je ale pro oba dva notové zápisy stejný.



Obrázek 42. Notový zápis písničky v G dur [66, s. 36]

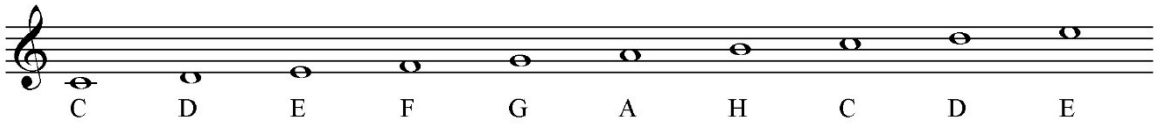
Notový zápis je ale možné použít i jako materiál pro výklad a procvičení kódování. Melodii je totiž možné zapsat číselným kódem v podobě dvojice čísel pro každou notu, přičemž první číslo udává výšku tónu a druhé jeho délku, viz Tabulka 4. Vytvoříme si tak vlastní, jednodušší obdobu MIDI - původně standardu pro komunikaci mezi elektronickými hudebními nástroji, s nímž dnes pracují i počítačové hudební programy.

Tabulka 4. Podklady pro kódování notového zápisu

Tón	Kód	Délka	Kód
C	1	𝅗𝅥	2
D	2	♩	4
E	3	♪	8
F	4		
G	5		
A	6		
H	7		

	Komprese
0	začátek opakování
9	konec opakování

Obrázek 43 přináší přehled názvů not v notové osnově.



Obrázek 43. Přehled názvů not

Zápis kódu podle prvního notového zápisu bude mít následující podobu:

18 28 38 48 54 54 64 64 52 64 64 52 48 48 48 48 34 34 24 24 52 48 48 48 48 34 34 24 24 12

Vzhledem k tomu, že některé kratší pasáže se opakují, lze vytvořit komprimovanou podobu, při jejímž zápisu využijeme neobsazená čísla 0 a 9 pro určení tónu – nulu pro označení začátku opakované pasáže a devítku pro označení konce opakované pasáže. V komprimované podobě pak bude zápis kódu vypadat následovně:

18 28 38 48 54 54 0 64 64 52 9 0 48 48 9 34 34 24 24 52 0 48 48 9 34 34 24 24 12

Zápis kódu podle druhého notového zápisu bude mít následující podobu:[59]

58 68 78 18 24 24 34 34 22 24 24 22 18 18 18 18 74 74 64 64 22 18 18 18 18 74 74 64 64 52

Nyní si, pro lepší přehled, srovnáme komprimovaný, tj. kratší zápis první a druhé verze:

18 28 38 48 54 54 0 64 64 52 9 0 48 48 9 34 34 24 24 52 0 48 48 9 34 34 24 24 12

58 68 78 18 24 24 0 34 34 22 9 0 18 18 9 74 74 64 64 22 0 18 18 9 74 74 64 64 52

Při pozorném prostudování obou kódů zjistíme, že oba zápisy se liší vždy pouze na první pozici každé dvojice, a to zcela pravidelně: ve druhé verzi je hodnota první číslice vždy o 4 vyšší, pokud by součet přesahoval hodnotu sedm, pak je naopak číslo o 3 nižší.

Tento materiál je tak možné použít i pro výklad úvodu do šifrování, konkrétně pro seznámení s jednou z nejstarších a zároveň nejjednodušších šifer. Zápis druhé verze při srovnání s první totiž připomíná tzv. *Caesarovu šifru*, u níž se každé písmeno v abecedě nahradí písmenem, které následuje v abecedě o *k* písmen dále. [67, s. 521]. V našem případě se $k = 4$.

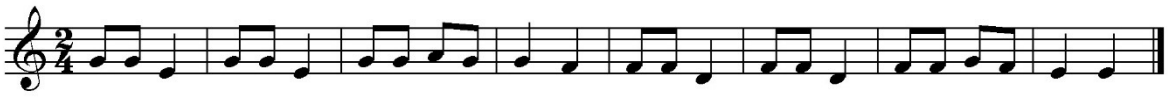
Písmena označující názvy not byla sice nahrazena čísly, k nimž byla přičítána nějaká konstantní hodnota, nikoliv jinými písmeny, ale pokud bychom text šifrovali pomocí

[59] Zmiňovaný jiný takt zápisu, nemá na zápis kódu, který se zaměřuje pouze na výšku a délku tónu, žádný vliv.

počítače,[60] postupovali bychom právě tak: písmenům bychom přiřadili číselný kód (např. podle tabulky ASCII) a k této číselné hodnotě bychom přičetli nějakou konstantu, vzniklé číslo bychom převedli zpět na písmeno[61] a takto zašifrovaný text by byl odeslán. Při dešifrování by se postupovalo analogicky.[62]

9.3.1 Cvičení

Žák zapíše melodii písničky *Skákal pes*, viz Obrázek 44, podle kódovací tabulky, viz Tabulka 4, v nekomprimované i komprimované verzi.



Obrázek 44. Notový zápis k úloze ke kódování

9.3.2 Řešení

Nekomprimovaná verze:

58 58 34 58 58 34 58 58 68 58 54 44 48 48 24 48 48 24 48 48 58 48 34 34

Komprimované verze:

0 58 58 34 9 58 58 68 58 54 44 0 48 48 24 9 48 48 58 48 34 34

9.3.3 Poznámky z praktické výuky

Spojení výkladu k modelům, kódům a kompresi na jednom výukovém materiálu se při praktickém testování ukázalo jako vhodné, pouze zmínka o šifrování, konkrétně o Caesarově šifře a jejím principu bude vhodnější v případě, že žáci už tuto látku znají. Pokud je pro ně toto téma nové, bude vhodnější tuto zmínku vynechat a tento materiál použít (jak je ostatně zmíněno) výše na úvod samostatné vyučovací hodiny věnované šifrování a Caesarově šifře, která je zpravidla první šifrou, s níž jsou žáci seznamováni.

---

[60] Bylo to mj. strojové dešifrování vojenských zpráv během 2. světové války, které urychlilo vývoj elektronické výpočetní techniky [23, s. 168] a které výrazně zkrátilo válku a ušetřilo mnoho lidských životů. [23, s. 171]

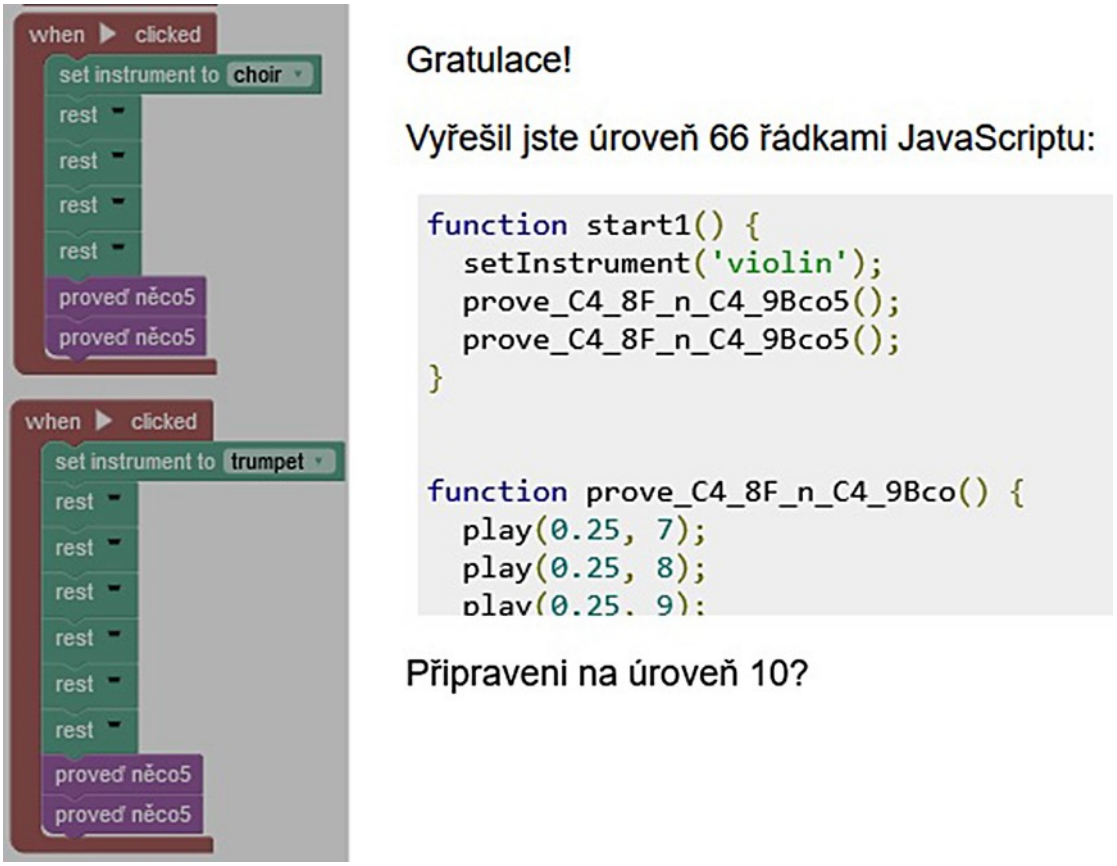
[61] U většiny programovacích jazyků jde o funkce *ord()* pro převod znaku na číselnou hodnotu a *chr()* pro převod číselné hodnoty na odpovídající znak.

[62] Pro šifrování melodie by ale tento postup nefungoval, protože po posunutí o 4 tóny je melodie v tomto případě naprosto shodná s původní, pouze je posunuta (o interval čisté kvinty) výš.

Podobně jako v případě převodu binárního kódu odvozeného z obrázku do desítkové soustavy, viz podkapitola 7.2.1, byla při praktickém testování žákům 6. a 7. ročníku pro samostatné kódování dána k dispozici nejen kódovací tabulka, viz Tabulka 4, ale také přehled názvů not, viz Obrázek 43.

Na webových stránkách *Blockly Games* v sekci *Music* lze v devíti krocích sestavit program, který ve čtyřhlase zahraje francouzskou lidovou písničku u nás známou jako *Bratře Kubo* jako kánon. Po zvládnutí každé dílčí úlohy systém zobrazí okno, v němž mj. ukáže, jak by daný program vypadal v zápisu zdrojového kódu v programovacím jazyku JavaScript.

Jak ukazuje Obrázek 45, noty jsou zde kódovány přesně v opačném pořadí, napřed délka a potom výška tónu. U výšky tónu se liší číslování (tón C zde má číslo 7, zatímco v předchozím případě měl číslo 1) a délka tónu je zapsána jako desetinné číslo, zatímco v předchozím případě byla délka tónu kódována jako jmenovatel zlomku, který vyjadřuje hodnotu desetinného čísla zapsaného v syntaxi jazyku JavaScript.



Obrázek 45. Výpis potvrzující vyřešení zadaného úlohy [68 a vlastní]

Je škoda, že systém neumožňuje zápis celých volných taktů pomocí bloku *Opakuj x-krát*, jako je tomu v sekci Želva. Tento způsob zápisu – viz Obrázek 45 opakované zelené bloky

rest – rozhodně nevede mladé začínající programátory ke vštípení si zásady, pro algoritmizaci jedné z nejzásadnějších, a sice DRY - Don't repeat yourself (neopakuj se).[63]

[63] Rovněž u zdrojového kódu zapsaném v JavaScriptu jsou opakované volné takty řešeny opakováním zápisu *rest(1);*

10 NÁROČNĚJŠÍ ÚLOHY Z ALGORITMIZACE

Závěrečná kapitola bude věnována několika náročnějším úlohám z algoritmizace, které jsou dostupné volně na internetu a k nimž není k dispozici řešení.

Na webových stránkách projektu umimeto.cz je v sekci *Informatika – Tvorba programu* – v závěru sekce *Želví grafika* pět těžších úloh pro vykreslování fraktálů označených jako *Záludné*, které ale lze řešit krátkým programem (v nadpoloviční většině případů jde o 5 řádků[64]), jen je potřeba mít ty správné znalosti, zkušenosti z předchozích úloh a správný nápad, příp. vhodnou nápovědu. [69]

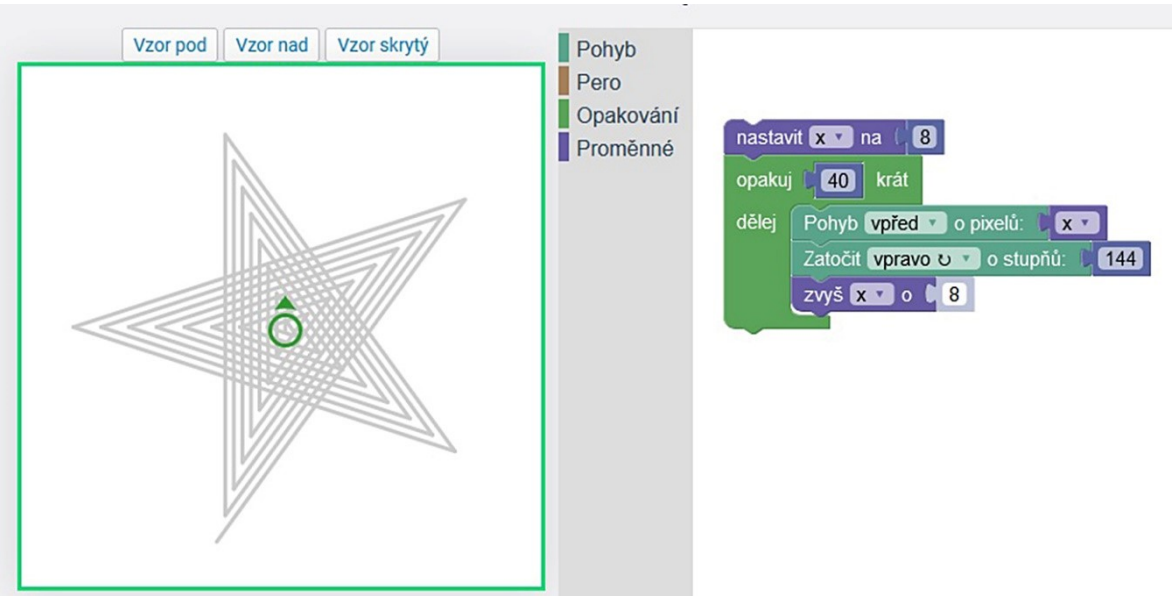
Tyto úlohy byly zpracovány v poněkud pozměněném pořadí, které lépe reflektuje jednu z nejzákladnějších pedagogických zásad, a sice zásadu stupňování obtížnosti, tj. postupovat od jednoduššího ke složitějšímu.

Celá sekce *Tvorba programu* končí několika úlohami „na hraní“, či přesněji řečeno „… *k experimentování*“, u nichž je již program připravený. Mezi těmito obrazci nechybí dva pravděpodobně nejznámější složitější fraktály Kochova vločka a Sierpinského trojúhelník.

10.1 Hvězdná spirála

Tato úloha, ačkoliv je autoři zařadili až jako čtvrtou v pořadí, je ze všech nejjednodušší. Nápověda v podobě již vložených příkazů říká, že výchozí hodnota *x* je 8, cyklus se opakuje 40krát a hodnota *x* se zvyšuje o 8. [70]

[64] I když délka zápisu zdrojového kódu nic nevypovídá o složitosti algoritmu v něm zapsaném.



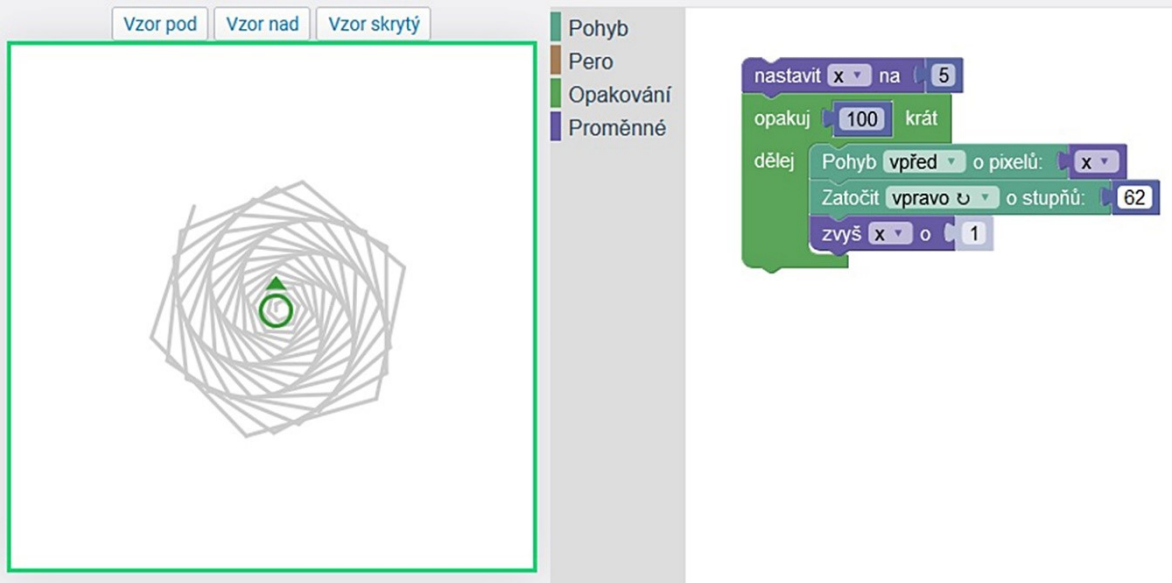
Obrázek 46. Hvězdná spirála – zadání a řešení [70 a vlastní]

10.1.1 Poznámky z praktické výuky

Žák doplní pouze dva příkazy pro pohyb, přičemž absolutní hodnotu 100 pixelů zamění za relativní proměnnou *x*, a otočení. Určit správný úhel otočení je na celé úloze asi to nejtěžší. Pokud ale žák ví, že vnitřní úhel pěticípé hvězdy je 36° (180 : 5), pak stačí žákům připomenout, že pro zadání vytvoření určitého úhlu je potřeba zadat hodnotu, která je rovna rozdílu mezi přímým úhlem (180°) a požadovaným úhlem, viz podkapitola 5.1.

10.2 Točící se spirála

Tuto úlohu zařadili autoři jako třetí, v našem případě je zařazena jako druhá nejlehčí, i když původní nápověda: „*Nejdřív zkuste zatáčet o 60 stupňů, pak zkuste mírně měnit úhel*.“ [71] může působit dojmem, že program pracuje se dvěma cykly a/nebo se dvěma proměnnými. Nápověda v již připravených příkazech říká, že výchozí hodnota *x* je 5, cyklus se opakuje 100krát a hodnota *x* se zvyšuje o jeden pixel. [71]



Obrázek 47. Hvězdná spirála – zadání a řešení [71 a vlastní]

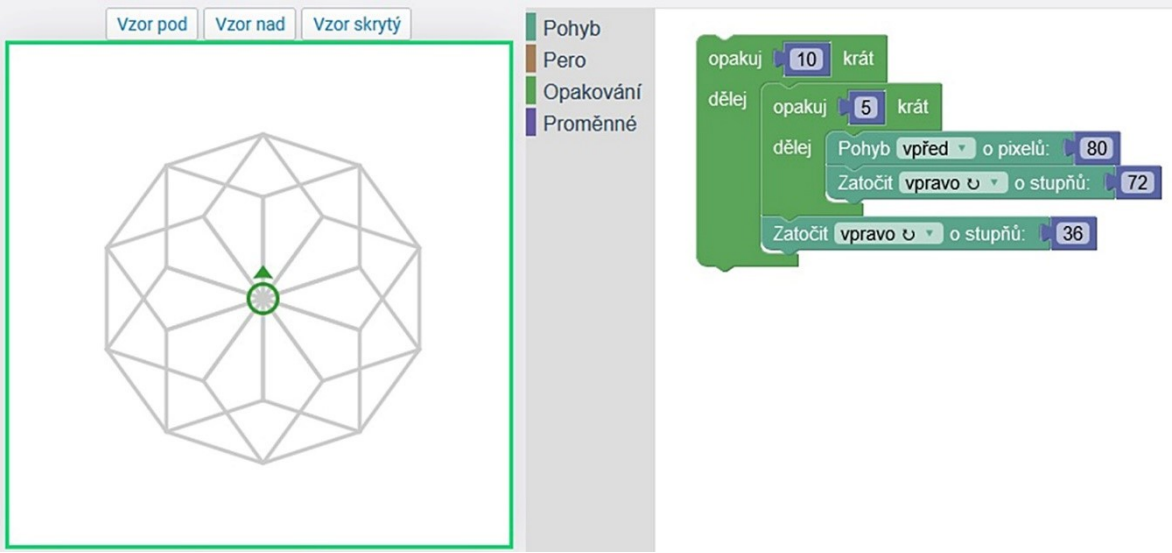
10.2.1 Poznámky z praktické výuky

Opět, stejně jako v předchozím případě, se jedná o pětiřádkový zdrojový kód, přičemž žák doplní pouze dva příkazy pro pohyb, a sice pohyb vpřed o proměnnou *x* a správný úhel zatočení, který musí hledat *metodou pokusu a omylu* (trial and error method).[65] Bude-li žák přidávat po jednotkách (v nápovědě je řečeno *mírně*), úspěšný bude již druhý pokus.

10.3 Diamant B

Rovněž u třetí úlohy, kterou autoři zařadili úplně na začátek (zřejmě jako z jejich pohledu nejlehčí) lze vystačit s pouhým pětiřádkovým kódem. Původní nápověda prozrazuje, že k řešení je potřeba použít pětiúhelníky. Nápověda v již vložených příkazech říká pouze, že délka strany pravidelného pětiúhelníku je 80 pixelů. [72]

[65] I tento postup je někdy součástí práce programátora, zejména při finálním jemném dolaďování programu.



Obrázek 48. Diamant B – zadání a řešení [72 a vlastní]

10.3.1 Poznámky z praktické výuky

Tato úloha je oproti dvěma předchozím mírně složitější, protože již obsahuje vnořený cyklus. Právě onen vnořený cyklus vykresluje pravidelné pětiúhelníky, jejichž délka je známa z nápovědy a velikost vnitřního úhlu by měla být žákům na 2. stupni ZŠ také známa. Úhel zatočení kreslícího nástroje, viz podkapitola 5.1 a Obrázek 16 je snadno odvoditelný z rovnice 360 : x, přičemž x udává počet vrcholů pravidelného mnohoúhelníku. V případě pětiúhelníku je to 72°, viz Tabulka 5.

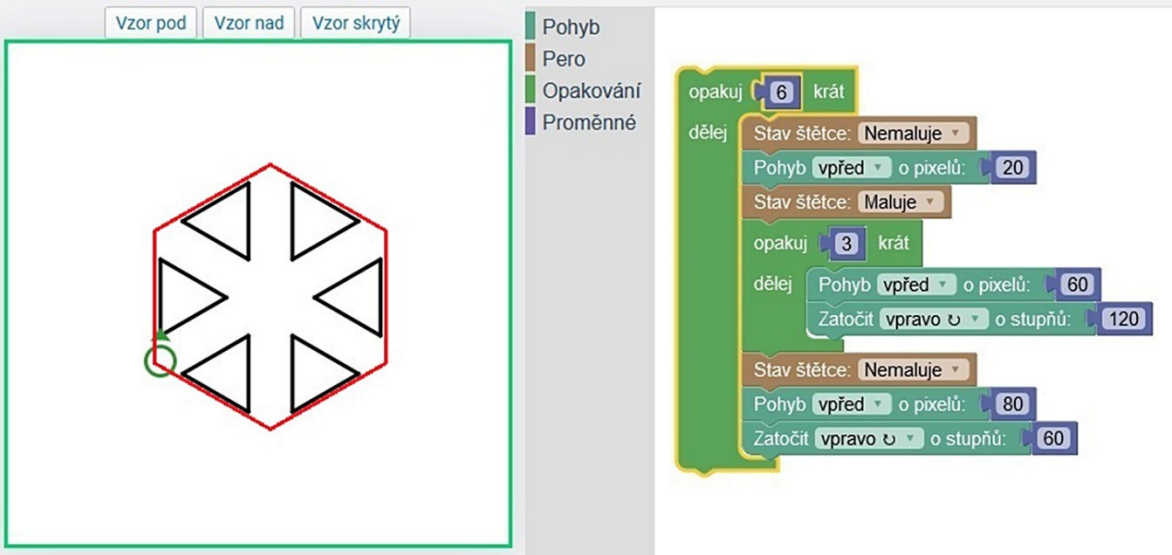
Tabulka 5. Úhly u pravidelných mnohoúhelníků [vlastní zpracování dle 43, s. 27]

	Počet úhlů	Součet úhlů	Velikost vnitřního úhlu	Úhel zatočení nástroje
Rovnoramenný trojúhelník	3	180°	60°	120°
Čtverec	4	360°	90°	90°
Pravidelný pětiúhelník	5	540°	108°	72°
Pravidelný šestiúhelník	6	720°	120°	60°
Pravidelný osmiúhelník	8	1 080°	135°	45°

Hlavní cyklus pak nechá nakreslit těchto pravidelných pětiúhelníků deset, přičemž po nakreslení každého pětiúhelníku natočí kreslící nástroj umístěný zpět přesně ve středu o polovinu hodnoty vnitřního úhlu pravidelného pětiúhelníku, tj. o 36°.

10.4 Šest trojúhelníků

Úloha nazvaná *Šest trojúhelníků*, autoři ji zařadili hned jako druhou, sice nepracuje s proměnnou (stejně jako předchozí úloha), ale, kromě toho, že se v ní vyskytuje vnořený cyklus, je zde potřeba také pracovat se zapínáním a vypínáním pera. Nápověda v podobě již vložených příkazů říká, že kreslící nástroj se bude pohybovat 20 px vpřed bez kreslení a následně 60 px rovněž vpřed, ovšem to již bude kreslit. [73]



Obrázek 49. Šest trojúhelníků – zadání a řešení [73 a vlastní]

10.4.1 Poznámky z praktické výuky

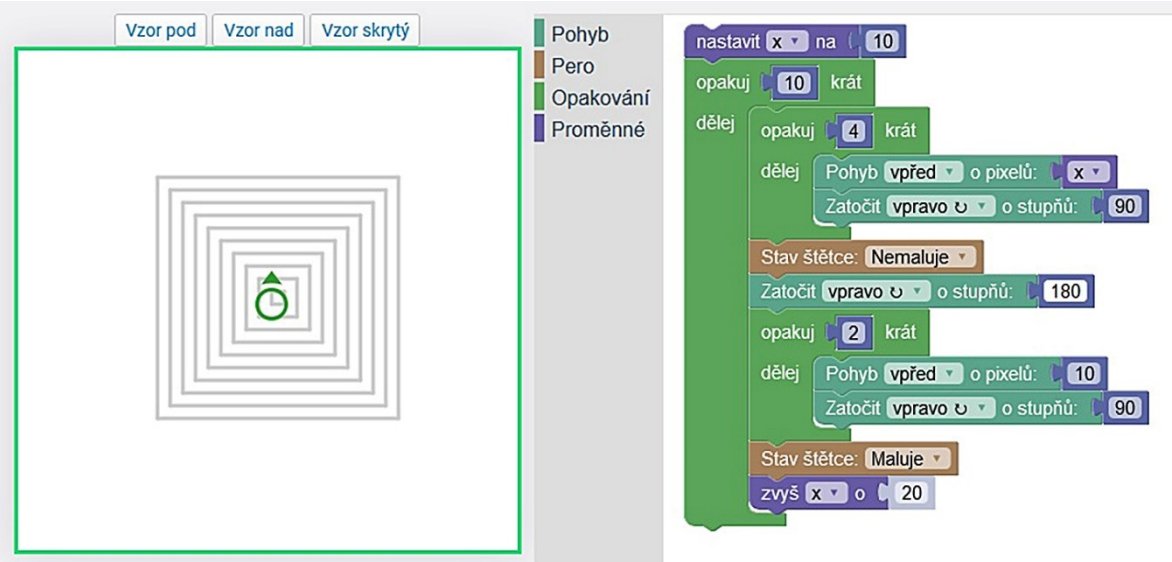
Pro řešení této úlohy je vhodné si představit kreslené trojúhelníky uvnitř pravidelného šestiúhelníku. Délka jeho stran je 100 px a vnitřní úhel svírá 120°. Zdrojový kód opět obsahuje vnořený cyklus, který nakreslí rovnostranný trojúhelník o délce strany 60 px.

Základní cyklus pak nakreslí těchto rovnostranných trojúhelníků šest, přičemž vždy po nakreslení jednoho trojúhelníku zvedne pero, přesune se o 80 px dopředu a otočí o 60° doprava. Poté začíná nový cyklus, na jehož začátku se stále ještě nekreslící pero posune novým směrem o 20 px. Pak teprve, po spuštění pera, začíná vykreslování nového trojúhelníku.

Pro žáky je poněkud matoucí, že zatímco u rovnostranného trojúhelníku, kde je potřeba vytvořit úhel o velikosti 60°, kreslící nástroj zatáčí o 120°, v případě šestiúhelníku, kde je potřeba vytvořit úhel o velikosti 120°, kreslící nástroj zatáčí o 60°.

10.5 Čtverce ve čtvercích

Tato úloha, která vytváří obrazec vyvolávající tzv. moaré efekt, zařadili na úplný závěr jako nejsložitější i autoři projektu umimeto.cz. U této úlohy bude využita práce s proměnnou, hned dva po sobě následující vnořené cykly a zvedání a pokládání pera. Veškerá nápověda konstatuje pouze to, že „… *mezery mezi čtverci mají velikost 10*.“ [74]



Obrázek 50. Čtverce ve čtvercích – zadání a řešení [74 a vlastní]

10.5.1 Poznámky z praktické výuky

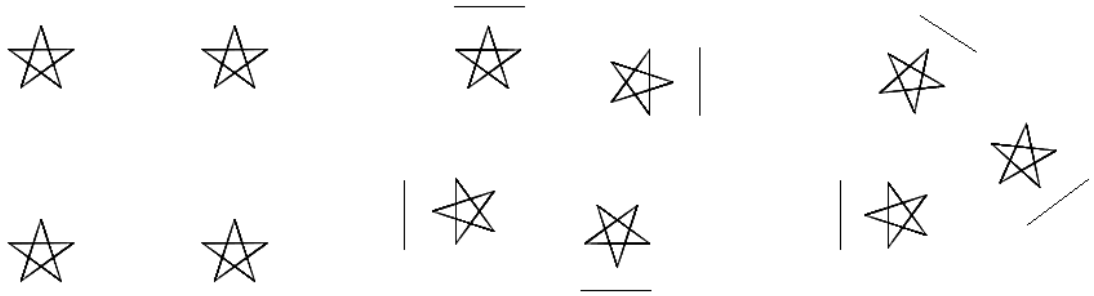
Co se týče výchozí délky strany čtverce, z předlohy je zřejmé, že odpovídá právě oněm 10 px, tj. rozměru mezery mezi čtverci. Délka strany následujícího čtverce je zřejmá: aby byl každý další čtverec vždy o 10 px z každé strany větší, musí být jeho strana vždy o 2 × 10 px, tj. o 20 px delší.

V případě přesunu kreslícího nástroje po dokončení předchozího čtverce na novou pozici, se nabízí varianta otočení o 135° vlevo, úhlopříčný přesun a nové otočení o 135° tentokrát vpravo. Délka úhlopříčného přesunu, podle Pythagorovy věty, je odmocnina hodnoty 200, což je iracionální číslo s nekonečným vývojem. Program ovšem funguje s hodnotou vyjádřenou např. na dvě desetinná místa 14,14 - samozřejmě psáno s desetinnou tečkou namísto desetinné čárky.

Řešení s úhlopříčným přesunem si vystačí pouze se dvěma cykly. Obrázek 50 na 8.-10. řádku[66] ukazuje řešení přesunu po odvěsnách rovnoramenného pravoúhlého trojúhelníku, nikoliv po přeponě, jak je popsáno v předchozím odstavci.

10.6 Hvězdy

Zbylé příklady pochází z webových stránek *Blockly Games*, které rovněž nenabízí řešení, a jen velmi strohou nápovědu k řešení úloh. Úloha číslo 5 v sekci Želva ukládá nakreslit čtyři pěticípé hvězdy uspořádané do čtverce. Čtverec, který naznačují hvězdy, ale není ve vodorovné poloze, což na první pohled vyvolává dojem, že tato skutečnost činí tuto úlohu obtížnější. Ve skutečnosti je mnohem obtížnější najít algoritmus a napsat zdrojový kód, který nakreslí hvězdy tak, jak ukazuje Obrázek 51 vlevo.



Obrázek 51. Hvězdy – Blockly Games, Želva, úlohy 5 a 6

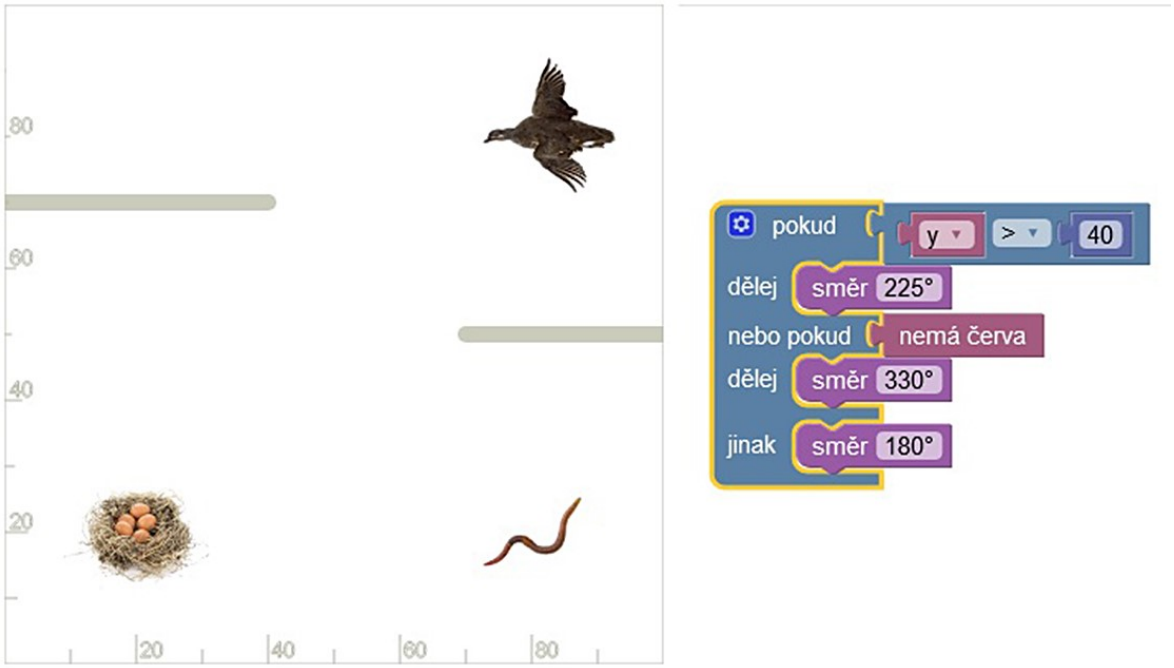
10.6.1 Poznámky z praktické výuky

Pro nalezení řešení je potřeba si všimnout, že vždy jedna z pěti čar hvězdy je buď svislá, nebo vodorovná a naznačuje tak čtverec, jak ukazuje Obrázek 51 uprostřed. Následující úloha, úloha číslo 6, je velmi podobná: zadání říká, aby byly hvězdy uspořádány do trojúhelníku. Stačí tedy změnit počet cyklů a změnit úhel otočení tak, aby vznikl úhel, který odpovídá velikosti vnitřního úhlu u rovnostranného trojúhelníku.

[66] Zápis zdrojového kódu bohužel nemá číslované řádky, což je pro orientaci v nich při jejich rozboru poněkud nevýhoda.

10.7 Pták

Na webové stránce *Blockly Games* sekce *Pták* předchází sekci *Želva*, ale je otázkou, zda jsou úlohy v této sekci lehčí než v sekci *Želva*. O z pedagogického hlediska nevhodném výběru a uspořádání jednotlivých úloh ve smyslu odpovídajícího a přiměřeného stupňování náročnosti předkládaných úloh již byla (a sice v podkapitole 2.6.2) a ještě bude řeč (v následující podkapitole).



Obrázek 52. Blocky Games, Pták 7, situace a řešení [75 a vlastní]

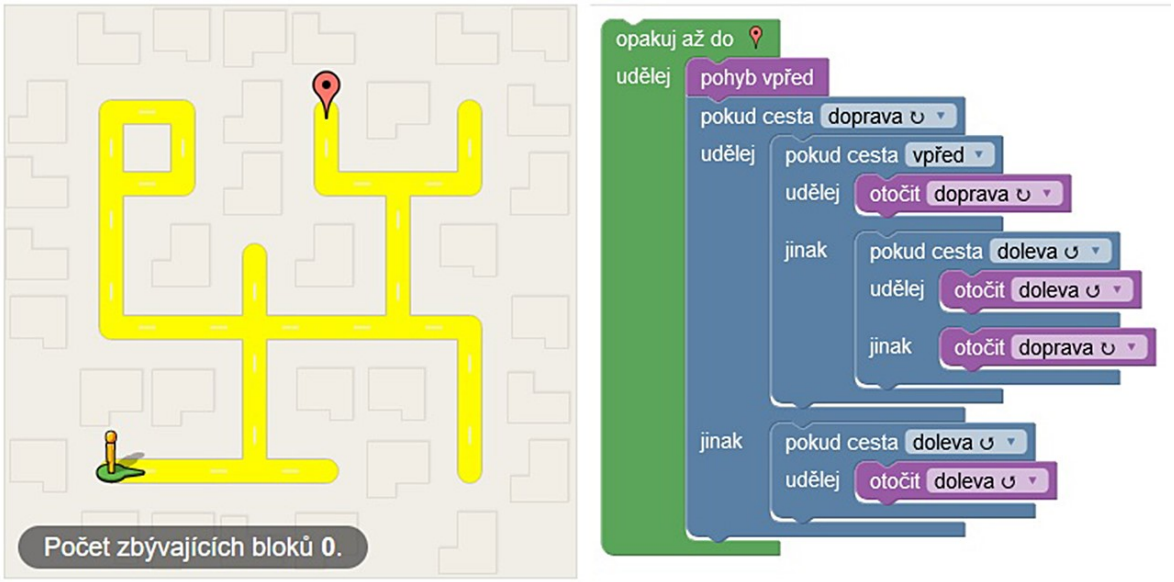
10.7.1 Poznámky z praktické výuky

Všechna předchozí řešení v této sekci, pokud byla v nabídce nástrojů podmínka „*nemá červa*“, začínala právě touto podmínkou. V tomto případě je tomu jinak. Pták musí letět směrem dolů pod určitým úhlem (který je potřeba hledat metodou pokusu a omylu) tak či tak, protože červ i hnízdo se nachází dole, zatímco on je nahoře. O dalším směru se bude rozhodovat v úrovni $y = 40$, a pokud chytne červa, otočí se směrem k hnízdu. S hodnotou x, tj. s vodorovnými souřadnicemi není vůbec potřeba pracovat.

Zbylé tři úrovně v této sekci jsou již obtížnější, protože pracují s kombinováním podmínek, tj. logickým operátorem AND.

10.8 Bludiště

Sekce *Bludiště* je hned v pořadí druhá sekce za sekcí *Skládačka*, která je opravdu velmi jednoduchá a evidentně slouží pouze k tomu, aby se žák naučil pracovat s bloky v podobě puzzle dílků (pokud ještě nemá tuto zkušenost). Prvních 9 úloh je poměrně velmi jednoduchých. Poslední desátá úloha je ovšem mimořádně obtížná a sami tvůrci upozorňují, že je určená pro velmi zkušené programátory.



Obrázek 53. Blocky Games, Bludiště 10, situace a řešení [76 a vlastní]

10.8.1 Poznámky z praktické výuky

Úloha je ztížena omezením počtu použitých bloků na 10, nebo absencí logického operátoru AND apod. Navíc řešení pracuje s podmínkami vnořenými až do třetí úrovně, zatímco doposud si předchozí řešení vystačila s podmínkami v jedné úrovni. Právě nemožnost použití logického operátoru AND si žádá nutnost vkládat podmínky do podmínek a vytvářet tak vícero úrovní. „*Jednou z nejdůležitějších zásad správného programování je povinnost psát ... programy maximálně přehledně*.“ [53, s. 50] Je otázkou, zda pro začátečníka je přehlednější delší verze (více řádků) s vnořenými podmínkami nebo kratší verze (o 3 řádky méně) s použitím logického operátoru AND. Z pohledu počítače (kompilátoru) jsou oba zápisy rovnocenné, tj. stejně funkční.

Tak či onak se tato úloha svojí obtížností jeví jako pro žáky základní školy naprosto nevhodná, protože zcela odporuje pedagogické zásadě přiměřenosti.[67]

10.9 Shrnutí kapitoly

Autoři projektu umimeto.cz jsou známí svým pedagogickým přístupem popsaným v podkapitole 6.2, a sice, že z pedagogického hlediska je nejcennější, když si žák přijde na řešení problému sám. Proto řešení neuvádí. Podobně zřejmě uvažují i autoři projektu *Blockly Games*, kde také není přímo k dispozici řešení a nápověda mnohdy není dostatečná. Druhým důvodem, proč řešení není uvedeno, je ten, že mnohdy neexistuje pouze jedno řešení a i zdrojový kód pracující se stejným algoritmem může být zapsán různým, i když velice podobným, způsobem.

Z úloh na *Blockly Games* zvládli žáci úlohy bez nápovědy či výraznější pomoci učitele po řešené případy, tj. u Bludiště po 9. včetně, u *Ptáka* po 6. včetně a u *Želvy* po 4. včetně. Uvedené obtížnější úlohy se při praktickém testování ukázaly jako méně vhodné pro hromadnou výuku, protože někdo je rychlejší a někdo pomalejší a správné řešení se dozví příliš brzy. Dříve, než by na něj přišel sám. Úlohy podobného typu, tj. s vyšším stupněm obtížnosti, se jeví jako vhodnější spíše pro domácí studium.[68]

Nevýhodou domácího samostudia je ale absence učitele, který by poradil a správně navedl, pokud si žák neví rady. V případě domácího samostudia je ale možné časově rozvolnit hledání řešení i na několik dní a využít tak, kromě možnosti konzultace s učitelem třeba druhý den ve škole, přínosu spánku a efektu „pasivního“ vyřešení problému, kdy člověka napadne řešení v okamžiku, kdy na něj zrovna soustředěně nemyslí. Obojí bylo rovněž popsáno v podkapitole 6.2.

[67] Zásada přiměřenosti vlastně doplňuje zásadu postupování od jednoduššího ke složitějšímu. Postupný nárůst obtížnosti by měl být přiměřený.
[68] Úloha *Bludiště* je pro svoji obtížnost nevhodná pro žáky základní školy i pro domácí studium.

ZÁVĚR

V metodice výuky základů informatiky, a zejména programování, se lze „*… setkat se dvěma protichůdnými přístupy. První … vychází od popisu funkce počítače, reprezentace dat v něm a konkrétního programovacího jazyka. Teprve když má …*.[žák] *‚jasno' ve všech těchto pojmech, přechází se k řešení úloh. Velkou nevýhodou tohoto postupu je, že …* [žák] *je od samého začátku zavalen velkým množstvím technických detailů, ve kterých se snadno může ztratit. Navíc bývá někdy úvod takového kursu dost nezáživný a mnoho …* [žáků] *odradí. Další negativní vlastnost se projeví až ‚po letech': takto vychovaní programátoři se většinou hůře vyrovnávají se změnami světa kolem nich (počítačů, programovacích jazyků, atd.).*" [24, s. 11]

„*Druhý přístup vykládá především algoritmizaci úloh a zcela abstrahuje od vlastností konkrétních počítačů i programovacích jazyků. … Největším nebezpečím tohoto přístupu je, že před …* [žákem] *zcela zatajuje technické problémy, které při psaní programů vznikají, a navíc často zbytečně oddaluje fázi, kdy je ….* [žák] *schopen psát vlastní programy.* [24, s. 11] Velký posun vpřed ve vývoji mladého programátora představuje okamžik, kdy si začne vymýšlet vlastní (programátorské) problémy a hledat (a postupně také nacházet) jejich (algoritmická) řešení. [77, s. 65] Programátoři, při jejichž výuce se psaní vlastních programů příliš oddalovalo, „*…mají často tendenci podceňovat konkrétní problémy a neradi dovádějí své projekty do konce*." [24, s. 11]

„*Nevýhody obou těchto přístupů jsou velmi dobře známy, proto se s nimi v ‚čisté' podobě již prakticky nesetkáte. Přesto však …* [lze] *s nevelkou nadsázkou rozdělit všechny kursy programování do dvou skupin, podle toho, ke kterému extrému se více blíží. Ty kursy, o nichž to nelze říci, mají většinou pouze nevýhody obou*." [24, s. 11]

Aktuální trend v didaktice výuky informatiky na základních školách inklinuje spíše ke druhému přístupu, tj. abstrahuje od vlastností počítačů, nezabývá se tolik hardwarem (HW). Důvodů je hned několik: kromě výše zmíněných nevýhod prvního přístupu je tu i fakt, že vývoj HW pokročil tak daleko, že začátečníky v oboru programování nijak nelimituje[69] (platí to i pro standardní počítače pořízené do škol původně pro výuku ICT) a že fungování

[69] např. kapacita operační paměti apod.

moderního HW je dnes již natolik složité, že tato látka by byla pro žáky ZŠ (zbytečně) příliš obtížná (např. symetrický a asymetrický multiprocesing apod.)

Učivo týkající se tématu HW se tak zpravidla zaměřuje spíše na periferie a jejich základní dělení na vstupní a výstupní.

Dalším trendem v didaktice informatiky, konkrétně tematického celku "Algoritmizace a programování", který lze aktuálně pozorovat, je zaměření na cykly[70] a upozadění druhého nejzákladnějšího nástroje, který má, vedle cyklů, programátor k dispozici, a tím jsou podmínky. [16, s. 95]

Tato práce, která si mj. klade za cíl inspirovat učitele informatiky, zvláště ty, kteří mají k dispozici pouze základní vybavení pro výuku ICT, čerpá mj. i ze starších zkušeností ze Spojených států amerických a z novozélandské koncepce CSunplugged - projektu výuky základů informatiky pro děti bez počítače.

V teoretické části se práce zabývá důležitostí pomůcek ve výuce (jimiž, jak bylo řečeno výše, nemusí být, a to ani při výuce informatiky, vždy pouze počítač) a představuje dostupné sbírky příkladů pro výuku informatiky na ZŠ, jak v papírové, tak v elektronické formě. Dále přináší základní výklad k algoritmizaci, která spolu s programováním tvoří v RVP pro ZV jeden ze dvou nových tematických celků, včetně několika (s tématem souvisejících) historických zajímavostí.

Další kapitoly teoretické části stručně pojednávají o kódování a modelování, představují želví grafiku (včetně jejích současných podob), která tvoří základ výuky programování již od 70. let minulého stol. Poslední kapitola teoretické části se vrací k obecnější pedagogice, a sice zejména ke kompetenci k řešení problémů, která je rovněž zahrnuta v aktuálním RVP jakožto jedna ze sedmi klíčových kompetencí.

První kapitola v praktické části je inspirována novozélandskou koncepcí CSunplugged, konkrétně jejím pojetím výkladu dvojkové soustavy a kódování. Nad rámec výchozího materiálu byla úloha, která měla prověřit zvládnutí látky, při praktické zkoušce ve 4. ročníku

[70] z nich pak především na tzv. *for*-cykly, u nichž předem známe počet opakování, před tzv. *while*-cykly, u nichž přesný počet opakování předem neznáme a u nichž může dojít (v důsledku programátorské chyby) k nekonečnému zacyklení

ZŠ rozšířena (ztížena) o převod dvojkového kódu, který byl odvozen z obrázkového kódu, do desítkového zápisu čísla.

Základní poznatky z praktické výuky jsou vždy stručně shrnuty v závěrečných podkapitolách nebo ve shrnutí závěrečné kapitoly v Praktické části.

Podobně bylo mírně ztíženo i další cvičení, které vycházelo z výše zmíněné novozélandské koncepce. Praktická úloha z oblasti rastrové grafiky byla sice již v původním materiálu rozšířena o kompresi dat, ale pro praktické testování byla tato úloha ještě mírně ztížena o nutnost dopočítávat chybějící pixely na konci řádku do celkového počtu pixelů na řádku.

Další příklad spojuje téma modelování, kódování, komprese dat, přičemž se krajně dotýká i tématu šifrování. Vše je vysvětleno a následně ověřeno, zda bylo správně pochopeno, na stejném výukovém materiálu, takže žáci jasně vidí rozdíl v technikách pracovního postupu u jednotlivých metod. Příklad je inspirován učebnicí *Základy informatiky pro 2. stupeň ZŠ* autorů Berki a Drábková, kde je ovšem daný materiál použit pouze pro výklad modelování. Toto rozšíření původního zadání bylo testováno v 6. a 7. ročníku ZŠ.

Následující kapitola je věnována třídění, které je nejen zcela „… *zásadní pro práci s téměř jakýmikoli informacemi*“ [32, s. 72], ale je také „…*jedním ze základních stavebních kamenů chodu počítače*.“ [32, s. 71]

Navíc třídící, zvané také řadící, algoritmy jsou dobrým konkrétním příkladem při (pozdějším) výkladu složitosti algoritmu.[71] Správné pochopení principů fungování jednotlivých třídících algoritmů je tak důležitou podmínkou pro (pozdější) pochopení problematiky složitosti algoritmů. Toto téma se jeví jako vhodné pro tzv. aktivizační výukové metody – aktuální trend v moderní pedagogice, konkrétně pro názorně demonstrační metodu, v níž je žák sám „… *aktivním činitelem celého procesu*.“ [65, s. 55] Bylo prakticky testováno ve 4. ročníku ZŠ.

Těžiště tvorby výukových materiálů (nejen pro informatiku) se stále více přesouvá do online prostředí s interaktivními výukovými programy. Z materiálů v papírové formě patří k „nejmladším“ *Programátorská cvičebnice* Radka Pelánka z roku 2012. Ta, kromě toho, že

[71] Třídící algoritmy lze použít i jako konkrétní příklady při výkladu dělení algoritmů; BubbleSort a InsertSort jsou příkladem iterativních algoritmů, zatímco MergeSort a QuickSort jsou příkladem rekurzivních algoritmů. [78]

obsahuje řešení alespoň těch nejtěžších ze zhruba 55 úloh (v ostatních případech je uvedena výrazná nápověda), je určena spíše pro střední a vysoké školy. Podobně k úlohám, tentokrát určeným široké škále věkových kategorií, na webových stránkách ibobr.cz, lze na samotných stránkách snadno dohledat nejen řešení jako takové, ale i výklad k němu a vysvětlení, co má daná úloha společného s informatikou.

Poslední kapitola této práce představuje pět závěrečných těžších úloh pro vykreslování fraktálů pomocí želví grafiky označených jako *Záludné* z webových stránek umimeinformatiku.cz a čtyři úlohy z webových stránek *Blockly Games*, kde není uvedeno řešení, pouze někdy méně, někdy více dostatečná nápověda. K těmto úlohám byly, kromě toho, že byly, podle jedné z nejznámějších pedagogických zásad, seřazeny do jiného pořadí od nejjednodušší po nejobtížnější, vypracovány poněkud obsáhlejší (než pouze původní strohá nápověda) praktické poznámky a závěrečné shrnutí kapitoly.

SEZNAM POUŽITÉ LITERATURY

[1] BERTRAND, Yves. *Soudobé teorie vzdělávání*. Praha: Portál, 1998. ISBN 80-7178-216-5.

[2] BROOKSHEAR, Glenn J. *Informatika*. Brno: Computer Press, 2015. ISBN 978-80-251-3805-2.

[3] *Rámcový vzdělávací program pro základní vzdělávání*. 2023. Online. Praha: MŠMT ČR, 2023. Dostupné z: https://www.edu.cz/rvp-ramcove-vzdelavaci-programy/ramcovy-vzdelavacici-program-pro-zakladni-vzdelavani-rvp-zv/ [cit. 2024-01-09]

[4] *Průmysl 4.0 a řízené utahování*. l. MAVE Plus, 2024. Dostupné z: https://www.prumyslove-naradi.cz/clanek/prumysl-4-0-a-rizene-utahovani-20 [cit. 2024-01-09]

[5] BOCKER, Hans J. *Svoboda jménem zlato: Vzpoura ve světě papírových peněz*. Praha: Austria Gold, 2001. ISBN 978-80-254-4979-0.

[6] TRUNEČEK, Jan. *Management v informační společnosti*. Praha: VŠE, 1997. ISBN 80-7079-201-9.

[7] Lenka Prokšová: *Školám chybí kvalifikovaní učitelé informatiky. Pracují raději v IT firmách*. Deník.cz. 22.1.2020. Dostupné z: https://www.denik.cz/z_domova/ucitele-k-pocitacum-chybeji-20200122.html [cit. 2024-01-12]

[8] RICHTEROVÁ, Bohdana; SEBEROVÁ, Alena; KUBÍČKOVÁ, Hana; SEKERA, Ondřej; CISOVSKÁ, Hana et al. *Akční výzkum v teorii a praxi*. Ostrava: Pedagogická fakulta Ostravské univerzity, 2020. ISBN 978-80-7599-176-8.

[9] ČUBA, František a HURTA, Josef. *Fungování podniků v současném světě*. Neubuz: Mondon, 2002. ISBN 80-903106-1-8.

[10] ZELENÝ, Jaroslav a MANNOVÁ, Božena. *Historie výpočetní techniky*. Praha: Scientia, 2006. ISBN 80-86960-04-8.

[11] FONTANA, David. *Psychologie ve školní praxi: příručka pro učitele*. Praha: Portál, 2014. ISBN 978-80-262-0741-2.

[12] KOLÁŘ, Zdeněk. *Výkladový slovník z pedagogiky: 583 vybraných hesel*. Praha: Grada, 2012. ISBN 97880247-3710-2.

[13] DOSTÁL, Jiří. *Metodické pomůcky a zásada názornosti*. Olomouc: Votobia, 2008. ISBN 978-80-7409-003-5. Dostupné z: http://mict.upol.cz/ucebni_pomucky_a_zasada_nazornosti.pdf [cit. 2024-01-10]

[14] CHROMÝ, Jan. *Materiální didaktické prostředky v informační společnosti*. Praha: Verbum, 2011. ISBN 978-80-904415-5-2.

[15] DOSTÁL, Jan. *Výukové programy*. Olomouc: Univerzita Palackého v Olomouci, 2011. ISBN 978-80-244-2782-9.

[16] HYLMAR, Radek. *Programování pro úplné začátečníky*. Brno: Computer Press, 2009. ISBN 978-80-251-2129-0.

[17] PECINOVSKÝ, Rudolf. *Začínáme programovat v jazyku Python*. Dotisk. Praha: Grada, 2021. ISBN 978-80-271-1237-1.

[18] ROHLÍKOVÁ, Lucie a VEJVODOVÁ, Jana. *Vyučovací metody na vysoké škole: praktický průvodce výukou v prezenční i distanční formě studia*. Praha: Grada, 2012. ISBN 978-80-247-4152-9.

[19] MONTESSORI, Maria. *Objevování dítěte*. Praha: Portál, 2017. ISBN 978-80-2621-234-8.

[20] POUSSIN, Charlotte. *Nauč mě, jak to mám udělat sám: pedagogika Montessori vysvětlená rodičům*. Praha: Svojtka & Co., 2018. ISBN 978-80-256-2459-3.

[21] MANĚNA, Václav. a kol. *Moderně s Moodlem: jak využít e-learning ve svůj prospěch?* Praha: CZ.NIC, 2015. ISBN 978-80-905802-7-5 (elektronická verze). Dostupné z: https://knihy.nic.cz/files/edice/moderne_s_moodlem.pdf [cit. 2024-01-19]

[22] JOBS, Steve. *The Next Insanely Great Thing*. Wired. Online. 1996, 12(2) ISSN 1078-3148 Dostupné z: https://www.wired.com/1996/02/jobs-2/ [cit. 2024-01-20]

[23] BELL, Tim; WITTEN, Ian H. a FELLOWS Mike. *CS Unplugged.* Christchurch: vydáno vlastním nákladem, 2015. nemá ISBN.

[24] DRÓZD, Januš a KRYL, Rudolf. *Začínáme s programováním*. Praha: Grada, 1992. ISBN 80-85-4244-1X.

[25] VELECKÝ, Jakub. *Dotyková zařízení a cloud ve výuce*. Online. 2014 Dostupné z: https://digifolio.rvp.cz/artefact/file/download.php?file=73639&view=11627 [cit. 2024-01-21]

[26] MACH, Jiří. *České školství za evropským průměrem stále více zaostává*. Novinky.cz. Online. 2023 Dostupné z: https://www.novinky.cz/clanek/veda-skoly-ceske-skolstvi-za-evropskym-prumerem-stale-vice-zaostava-40447831#dop_ab_variant=0&dop_source_zone_name=novinky.sznhp.box&source=hp&seq_no=1&utm_campaign=&utm_medium=z-boxiku&utm_source=www.seznam.cz [cit. 2024-01-22]

[27] *Modelové školní vzdělávací programy pro základní vzdělávání*. Online. České Budějovice: Jihočeská univerzita v Českých Budějovicích, 2018 Dostupné z: https://imysleni.cz/svp/svp-zv [cit. 2024-01-22]

[28] *Bobřík informatiky*. Dostupné z: https://www.ibobr.cz/ [cit. 2024-03-24]

[29] *Bebras - Thinking - History*. Online. Dostupné z: https://www.bebras.org/about.html [cit. 2024-03-24]

[30] *Bobřík informatiky – Věkové kategorie*. Online. Dostupné z: https://www.ibobr.cz/o-soutezi/vekove-kategorie [cit. 2024-03-24]

[31] CARLSON, Edward H. Teach Your Kids Programming. *Creative Computing*, 1983, 12(4), s. 168-176, ISSN 0097-8140. Dostupné z: https://archive.org/details/CreativeComputing198304/page/n167/mode/2up [cit. 2024-01-12]

[32] CHRISTIAN, Brian a GRIFFITHS, Tom. *Algoritmy pro život: jak využít počítačové algoritmy při každodenním rozhodování*. Brno: Jan Melvil Publishing, 2017. ISBN 978-80-7555-037-8.

[33] MAREŠ, Martin a VALLA, Tomáš. *Průvodce labyrintem algoritmů*. Praha: CZ.NIC, 2017. ISBN 978-80-88168-22-5 (elektronická verze). Dostupné z: https://knihy.nic.cz/files/edice/pruvodce_labyrintem_algoritmu_v2.pdf [cit. 2024-01-31]

[34] BERAN, Ladislav a ONDRÁČKOVÁ, Ivana. *Prověřte své matematické nadání*. Praha: SNTL, 1988. nemá ISBN.

[35] STOLL, Heinrich Alexander. *Sen o Tróji*. Praha: Svoboda, 1983. nemá ISBN.

[36] KLINE, Moris. *Mathematics for Nonmathematicians*. New York: Dover Publications, Inc., 1967. ISBN 978-04-8624-823-3.

[37] HOFSTADTER, Douglas R. *Gödel, Escher, Bach: existenciální gordická balada : metaforická fuga o mysli a strojích v duchu Lewise Carrolla*. Zip (Argo: Dokořán). Praha: Argo, 2012. ISBN 978-80-7363-265-6.

[38] HARDY, Godfrey Harold: *Obrana matematikova*. Praha: Prostor, 1999. ISBN 978-80-7260-544-6.

[39] ŠKRÁŠEK, Josef a TICHÝ, Zdeněk. *Základy aplikované matematiky*. Praha: SNTL, 1983. nemá ISBN.

[40] KLÍMA, Vlastimil. Eliptické křivky a šifrování. *CHIP: počítačový magazín*. Praha: Vogel Publishing 2002, 12(9), s. 134-136, ISSN 1210-0684.

[41] KLINE, Moris. *Mathematics And the Physical World*. London: John Murray, 1960. ISBN 978-0486241043. Dostupné z: https://archive.org/details/klinemathematicsandthephysicalworld/page/n7/mode/2up

[42] KOWAL, Stanislaww. *Matematika pro volné chvíle*. Praha: SNTL, 1985. nemá ISBN.

[43] ADLER, Irving. *Learning with Colours Mathematics: exploring the world of numbers and space*. Feltham: Paul Hamlyn, 1968. ISBN 978-06-0108-003-8.

[44] CRILLY. Tony. *Matematika: 50 myšlenek, které musíte znát*. Praha: Slovart, 2010. ISBN 978-8073-91-409-7.

[45] VIRIUS, Miroslav. *Základy algoritmizace*. Praha: ČVUT, Fakulta FJFI, 2008. nemá ISBN.

[46] ČADA, Ondřej. *Objektové programování: Naučte se pravidla objektového myšlení*. Praha: Grada, 2009. ISBN 9788024727455.

[47] STRELEC, Michal. *Co je to kódování?* strelec.pro. Online. 2024 Dostupné z: https://www.strelec.pro/slovnik-vyvojare/co-je-to/kodovani [cit. 2024-03-14]

[48] GÁLA, Libor; POUR, Jan a ŠEDIVÁ, Zuzana. *Podniková informatika: počítačové aplikace v podnikové a mezipodnikové praxi*. 3., aktualizované vydání. Management v informační společnosti. Praha: Grada Publishing, 2015. ISBN 978-80-247-5457-4.

[49] *Kódování a modelování*. Online. Umíme informatiku. Online. Dostupné z: https://www.umimeinformatiku.cz/cviceni-kodovani-modelovani [cit. 2024-01-31]

[50] TÖPFER, Pavel. *Algoritmy a programovací techniky*. Praha: Prometheus 1995. ISBN 80-85849-83-6.

[51] LARSEN, Sally Greenwood. *Computers for Kids*. 2. vyd. New Jersey: Creative Computing Press, 1981. ISBN 978-09-1668-822-6. Dostupné z: https://archive.org/details/ataribooks-coamputers-for-kids [cit. 2024-02-03]

[52] PELÁNEK, Radek. *Programátorská cvičebnice*. Brno: Computer Press, 2012. ISBN 978-80-251-3751-2.

[53] PECINOVSKÝ, Rudolf. *Python: kompletní příručka jazyka pro verzi 3.9*. Praha: Grada Publishing, 2020. ISBN 978-80-271-12-69-2.

[54] BLAHO, Andrej a KALAŠ, Ivan. *Imagine Logo: Učebnice programování pro děti*. Brno: Computer Press, 2006. ISBN 80-251-10-15-x.

[55] PILGRIM, Mark. *Ponořme se do Python(u) 3*. Praha: CZ.NIC, 2010. ISBN 978-80-904248-2-1 (elektronická verze). Dostupné z: https://knihy.nic.cz/files/edice/python_3.pdf [cit. 2024-02-09]

[56] ŠKODA, Jiří a DOULÍK, Pavel. *Psychodidaktika: metody efektivního a smysluplného učení a vyučování*. Praha: Grada, 2011. ISBN 978-80-247-3341-8.

[57] KOPKA, Jan. *Umění řešit matematické problémy*. Praha: HAV, 2013. ISBN 978-80-903625-5-0.

[58] ŠVAMBERK ŠAUEROVÁ, Markéta. *Techniky osobnostního rozvoje a dušení hygieny učitele*. Praha: Grada, 2018. ISBN 978-80-2475-255-6.

[59] PELÁNEK, Radek. *Šifry pro děti*. Praha: Edika 2022. ISBN 978-80-266-1715-0.

[60] CALICCHIO, Stefano. *Řešení problémů ve 4 krocích: Jak porozumět problémům a řešit je pomocí nejlepších strategií z psychologie a vědy o rozhodování*. Rakuten Kobo, 2027. ISBN 979-12-2206-916-6 (elektronická verze, paywall). Dostupné z: https://www.kobo.com/cz/cs/ebook/reseni-problemu-ve-4-krocich [cit. 2024-03-10]

[61] SKOUPÝ, Tomáš. *V Polsku zakážou na základních školách domácí úkoly*. Novinky.cz. Online. 2024 Dostupné z: https://www.novinky.cz/clanek/zahranicni-evropa-v-polsku-zakazou-na-zakladnich-skolach-domaci-ukoly-40458006#dop_ab_variant=1164111&dop_source_zone_name=novinky.sznhp.box&source=hp&seq_no=2&utm_campaign=&utm_medium=z-boxiku&utm_source=www.seznam.cz [cit. 2024-02-03]

[62] BERAN, Ladislav a ONDRÁČKOVÁ, Ivana. *Žádné obavy z matematiky: pomoc středoškolákům*. 2. vyd. Praha: SPN, 1986. nemá ISBN.

[63] TOMAN, Jiří. *Metody a technika informační činnosti*. Praha: SNTL 1970. nemá ISBN.

[64] ZIENTARA. Marguerite. *The History of Computing*. Framingham: CW Communication, Inc. 1981. nemá ISBN. Dostupné z: https://archive.org/details/TheHistoryOfComputing [cit. 2024-01-29]

[65] ZORMANOVÁ, Lucie. *Výukové metody v pedagogice: tradiční a inovativní metody, transmisivní a konstruktivistické pojetí výuky, klasifikace výukových metod*. Praha: Grada, 2012. ISBN 978-80-2474-100-0.

[66] BERKI, Jan a DRÁBKOVÁ, Jindra. *Základy informatiky pro 2. stupeň ZŠ*. Online Liberec: Technická univerzita v Liberci, 2020. ISBN 978-80-7494-521-2. Dostupné z: https://imysleni.cz/ucebnice/zakladyinformatiky-pro-zakladni-skoly. [cit. 2024-02-01]

[67] KUROSE, James F. a ROSS, Keith W. *Počítačové sítě*. Brno: Computer Press, 2014. ISBN 978-80-2513-825-0.

[68] *Blockly Games – Music 9*. Online. Dostupné z: https://blockly.games/music?lang=cs&level=9 [cit. 2024-03-25]

[69] *Umíme informatiku – Tvorba programu – Želví grafika*. Online. Dostupné z: https://www.umimeinformatiku.cz/zelvi-grafika [cit. 2024-03-18]

[70] *Umíme informatiku – Tvorba programu – Želví grafika – Hvězdná spirála*. Online. Dostupné z: https://www.umimeinformatiku.cz/zelvi-grafika-zaludne/43 [cit. 2024-03-18]

[71] *Umíme informatiku – Tvorba programu – Želví grafika – Točící se spirála*. Online. Dostupné z: https://www.umimeinformatiku.cz/zelvi-grafika-zaludne/22 [cit. 2024-03-19]

[72] *Umíme informatiku – Tvorba programu – Želví grafika – Diamant B*. Online. Dostupné z: https://www.umimeinformatiku.cz/zelvi-grafika-zaludne/15 [cit. 2024-03-18]

[73] *Umíme informatiku – Tvorba programu – Želví grafika – Šest trojúhelníků*. Online. Dostupné z: https://www.umimeinformatiku.cz/zelvi-grafika-zaludne/83 [cit. 2024-03-18]

[74] *Umíme informatiku – Tvorba programu – Želví grafika – Čtverce ve čtvercích*. Online. Dostupné z: https://www.umimeinformatiku.cz/zelvi-grafika-zaludne/23 [cit. 2024-03-19]

[75] *Blockly Games – Pták 7*. Online. Dostupné z: https://blockly.games/bird?lang=cs&level=7 [cit. 2024-03-25]

[76] *Blockly Games – Bludiště 10*. Online. Dostupné z: https://blockly.games/maze?lang=cs&level=10 [cit. 2024-03-27]

[77] PÓLYA, George. *How to solve it: a new aspect of mathematical method*. Dublin: Penguin books, 2022. ISBN 978-0-140-12499-6.

[78] *Algoritmy.net – Teorie Algoritmů – Algoritmus*. 2016. Online. Dostupné z: https://www.algoritmy.net/article/1240/Algoritmus [cit. 2024-04-11]

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

ASCII	American Standard Code for Information Interchange - americký standardní kód pro výměnu informací
CS	computer science – počítačová věda, informatika
DRY	Don't Repeat Yourself – neopakuj se
DSA	Digital Signature Algorithm - algoritmus digitálního podpisu
GSM	Groupe Spécial Mobile - celosvětově nejrozšířenější standard pro digitální mobilní sítě
HW	hardware - veškeré fyzické součásti počítače
ICT	Information and Communication Technology – informační a komunikační technologie
IDE	Integrated Development Environment – vývojové prostředí
IoT	Internet of Things – internet věcí
ISBN	International Standard Book Number – mezinárodní standardní číslo knihy
IT	Information Technology – informační technologie
LSb	Least significant bit – nejméně významný bit
MIDI	Musical Instrument Digital Interface – digitální rozhraní hudebního nástroje
MŠMT ČR	Ministerstvo školství, mládeže a tělovýchovy České republiky
PRIM	Podpora rozvoje informatického myšlení
px	Pixel – zkrácení anglických slov *picture element* - obrazový prvek (prvek obrázku)
RLE	run-length encoding – kódování podle délky
RSA	Rivest, Shamir, Adleman - šifra s veřejným klíčem
RVP	Rámcový vzdělávací program (curriculum framework)
ŠVP	Školní vzdělávací program
ZŠ	Základní škola
ZV	Základní vzdělávání

SEZNAM OBRÁZKŮ

SEZNAM TABULEK