UNIVERZITA PALACKÉHO V OLOMOUCI

PEDAGOGICKÁ FAKULTA

Katedra technické a informační výchovy

Diplomová práce

Bc. Vlastislav Novák

Tvorba webové aplikace pro podporu rozvoje informatického myšlení

Olomouc 2021

vedoucí práce: Mgr. Tomáš Dragon

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně a uvedl jsem v ní veškerou literaturu a ostatní informační zdroje, které jsem použil.

V Olomouci dne 2. 7. 2021 .....

vlastnoruční podpis

Poděkování

Rád bych poděkoval Mgr. Tomáši Dragonovi za cenné rady, věcné připomínky a vstřícnost při konzultacích a vypracování této diplomové práce.

Obsah

Úvod

V posledních letech dochází k zásadním změnám přístupu k výuce informatiky, ačkoliv Česká republika lehce pokulhává za západním světem. Opouští se zastaralé programovací jazyky, jakým je např. Pascal a do popředí se dostávají modernější a perspektivnější jazyky. Mění se také metody výuky, místo pouhé znalosti syntaxe programovacího jazyka se klade důraz na rozvoj tzv. informatického myšlení (v angličtině computational thinking). Jedná se o relativně nový pojem, který se dostal do širšího povědomí teprve v roce 2006. Velmi zjednodušeně se dá popsat, jako schopnost myslet jako informatik při řešení problémů. Tato schopnost je čím dál více ceněná a vyžadovaná. V současné přetechnizované době totiž nestačí zvládat práci s jednou konkrétní aplikací nebo programovacím jazykem, ale spíše umět se adaptovat na nové trendy v digitálních technologiích. Výhodou je, že tuto schopnost lze u dětí začít rozvíjet již na prvním stupni základních škol. Vždyť s algoritmy se tak jako tak všichni setkáváme v běžném životě.

Cílem této práce je tvorba webové aplikace pro podporu rozvoje informatického myšlení využitelné konkrétně v informaticky zaměřených kurzech v rámci volnočasových aktivit. První dvě kapitoly teoretické části se proto věnují volnočasovým aktivitám a zájmovému vzdělávání obecně. Podrobněji je popsána jedna konkrétní forma zájmového vzdělávání, která byla vybrána pro tuto práci jako ideální, a tou je počítačový kurz.

Třetí kapitola se snaží důkladně charakterizovat pojem informatické myšlení a jeho význam ve světě i v České republice. Snaží se definovat jeho současnou pozici ve výuce a hledá vhodné nástroje pro jeho rozvoj. Čtvrtá kapitola některé konkrétní nástroje dostupné na internetu analyzuje a vyhodnocuje nejvhodnější, podle předem definovaných kritérií.

Dále práce polemizuje nad vhodností výuky algoritmizace a programování. Z řad odborníků zní názory, že je potřeba vyučovat tvorbu algoritmů a jejich vyjádření v nějakém programovacím jazyce. Podle některých je dokonce programování v této oblasti nezbytné.

Bylo rozhodnuto, že vytvořená aplikace bude rozvíjet informatického myšlení skrze výuku programovacího jazyka Python. Ten je ideální pro výuku jedinců, kteří již základy algoritmizace mají za sebou, tak těch, kteří se s ní ještě nesetkali. Tuto myšlenku potvrzují průzkumy oblíbenosti, které jej řadí na první místo. Velkou měrou k tomu přispívá čistota a jednoduchost syntaxe, která se snaží být co nejsrozumitelnější.

Praktická část práce řeší tvorbu a popis aplikace. Specifikuje požadavky, popisuje metody a vývojové nástroje, technologie použité při vývoji a samotnou implementaci. Prezentuje finální verzi aplikace, její funkcionalitu a využití.

1 Volnočasové aktivity

Volnočasové aktivity jsou téma, které se ve společnosti dlouhodobě těší velkému zájmu. Volný čas a jeho trávení jsou důležitou součástí lidského života. V současné literatuře se lze setkat s mnohými definicemi, které se snaží pojem volný čas co nejpřesněji popsat. Autoři, kteří se této problematice věnují, zdůrazňují různé pojetí a také vnímání podstaty volného času.

1.1 Definice pojmu volný čas

Volný čas má mezi českými i zahraničními autory mnoho definicí, obecně lze však říci, že se jedná o svobodný čas, ve kterém si volíme činnosti dle vlastního uvážení. Můžeme jej vnímat jako čas odpočinku, regenerace, zábavy, či seberealizace v zájmových činnostech.

Hofbauer ve své publikaci Děti, mládež a volný čas vymezuje pojem volný čas jako „čas, kdy člověk nevykonává činnosti pod tlakem závazků, jež vyplývají z jeho sociálních rolí, zvláště z dělby práce a nutnosti zachovat a rozvíjet svůj život. Někdy se vymezuje jako čas, který zbývá po splnění pracovních i nepracovních povinností.“ (Hofbauer, 2004, s. 13)

Podle Hájka (2008, s. 10) je volný čas součástí života, který se nachází mimo pracovní a vázaný čas (biofyziologické potřeby, péče o rodinu a domácnost atd.). Je to doba, kdy se člověk věnuje činnostem, které ho rozvíjejí. Patří sem odpočinek a zábava, rozvíjení vlastních zájmů a koníčků, sebevzdělávání a zlepšování kvalifikace a také účast na veřejném životě.

V současnosti převažují dva pohledy, jakými můžeme nahlížet na volný čas. Jedná se o pozitivní a negativní vymezení. Předchozí uvedené definice odpovídají negativnímu vymezení volného času. V tomto případě volný čas v podstatě definujeme tím, co jím není. Z celkového času odečteme čas plnění povinností a biologických potřeb, a zbývající čas označíme jako volný čas. Jde tedy o čas, který nám zbyde po podmíněných činnostech. O samotném volném čase nám to však příliš neříká.

Proto je vhodnější si jej definovat opačným přístupem, tedy pozitivním vymezením. Pozitivní vymezení nám říká, čím volný čas je a čím je charakterizován. Kratochvílová (2004, s. 79) popisuje pozitivní vymezení volného času takto: „je to čas na oddych, rekreaci, regeneraci fyzických a psychických sil, uvolnění po práci, studiu, na společenská setkání […], na poznávání světa, života, na seberealizaci v aktivitách, činnosti podle vlastních potřeb a zájmů, představ, tužeb, aspirací a hodnot“.

Vážanský pozitivní vymezení definuje jako „dobu, která jedinci umožňuje svobodnou volbu činností, kdy se může svobodně realizovat a dělat to, co chce, k čemu jej nikdo nenutí

a k čemu také není podvědomě nucen.“ Volný čas podle něj lidem poskytuje spoustu možností, které mohou dělat, ale zároveň dělat nemusí. Ve volném čase si každý dle vlastních preferencí vybere, vyzkouší a vystřídá různé druhy aktivit.

Negativní vymezení volného času je tedy spíše časové a určuje, po jakých činnostech nastává. Naproti tomu pozitivní vymezení je obsahové a je především určeno jednotlivými aktivitami, jimiž čas vyplníme. Lze také říci, že negativní vymezení definuje činnosti, po kterých následuje volno, zatímco pozitivní ukazuje, k čemu můžeme volný čas využít.

1.2 Funkce volného času

Stejně jako definice volného času, i jeho funkce mají spoustu možných výkladů. Nejčastěji vymezujeme tyto 4 základní funkce (Bendl a kol., 2015, s. 122):

- **výchovně - vzdělávací**
- **zdravotní**
- **sociální**
- **preventivní**

Zařízení zabývající se výchovou ve volném čase přistupují k práci s těmito funkcemi různě, v závislosti na svém zaměření. Postupem času se význam funkcí měnil. První instituce se zabývaly péčí o děti po vyučování. Zásadní tedy bylo poskytnutí vhodného zázemí k zabránění nevhodným činnostem. Kladl se tudíž důraz hlavně na preventivní a sociální funkci. Později se začala stávat důležitější funkce výchovně-vzdělávací. Nadto bylo zjištěno, že při účelném působení na jedince měla i zdravotní, sociální a preventivní dopady. V současnosti je brána jako nejdůležitější právě výchovně-vzdělávací funkce (Hájek, 2008, s. 70).

Výchovně - vzdělávací funkce

Tato funkce je považována za prioritní, protože cílevědomě ovlivňuje osobnost jedince. Působí na všechny jeho složky osobnosti – fyzickou, psychickou, sociální i rozumovou. Podmínkou jejího úspěchu je správná volba pedagogických prostředků, které respektují specifika výchovy mimo výuku. Výchova mimo výuku umožňuje uspokojovat potřeby (např. seberealizace, rozšiřovat a prohlubovat zájmy, objevovat a rozvíjet speciální dovednosti jedince. V zájmových skupinách dochází k takovému sociálnímu učení, kdy jedinec nabývá nové vědomosti a dovednosti užitečné v interakci s ostatními, motivuje k smysluplnému využití

volného času, k osvojování nových dovedností a uvědomění si významu celoživotního vzdělávání. (Pávková, 2015, s. 122; 2014, s. 14-15; 2008, s. 70)

Zdravotní funkce

Zdravotní funkce je plněna zejména vhodným režimem, který pomáhá dodržovat zdravý životní styl, podporuje tělesný, duševní a sociální vývoj. Dobrý zdravotní stav lze podle Pávkové (2015, s. 122-123; 2014, s. 15; 2008, s. 70-71) podpořit těmito způsoby. Vhodným uspořádáním dne podle věku a individuálních specifik. Zařazením venkovních pohybových aktivit. Ty jsou důležité jakožto kompenzace dlouhého sezení ve vyučování. Cílem je, aby se tyto aktivity staly návykem a každodenní rutinou. Dále vedením ke správné životosprávě, tzn. zdravým stravovacím návykům a dodržování pitného režimu. Také jde o upevňování hygienických návyků, což se mimo jiné týká i odívání, péče o osobní věci atd. Důležité je i zvládání duševní hygieny, zvyšování odolnosti a podpora dobrých vztahů.

Sociální funkce

Význam této funkce spočívá v zajištění bezpečnosti žáků a jejich ohlídání po skončení vyučování, v době, kdy jsou rodiče v práci či jinak zaneprázdněni. V tomto smyslu ji zajišťuje školní družina. Dále sociální funkce obstarává vytváření nových sociálních vazeb, zaujímání nových sociálních rolí, zlepšování komunikačních dovedností a rozvíjení sociálních interakcí. Školní družina také srovnává rozdílné materiální a psychologické podmínky rodiny, a to věnováním stejné péče a pozornosti všem žákům, poskytnutím hraček, nástrojů a vybavení (Pávková, 2015, s. 123; 2014, s. 15-16; Pávková, Pavlíková a Hrdličková, 2002, s. 40).

Preventivní funkce

K základním oblastem prevence patří předcházení negativních sociálně patologických jevů. Těmi mohou být neposlušnost, lhaní, agresivita, útěky z domova, záškoláctví, kouření, alkoholismus, drogová závislost, šikana, intolerance, rasismus, xenofobie, kriminalita a delikvence apod. Školská zařízení podílející se na výchově mimo vyučování se zaměřují na primární prevenci určenou jedincům, kteří nejsou rizikoví (Pávková, 2015, s. 123; 2014, s. 16; 2008, s. 72). U jedince, který správně tráví volný čas a umí si vhodně zvolit volnočasovou aktivitu je méně pravděpodobné, že se u něj projeví některý z uvedených patologických jevů (Pávková, 2015, s. 123).

1.3 Požadavky na výchovu ve volném čase

Je potřeba dodržovat rozmanité požadavky, aby byla výchova ve volném čase úspěšná. Ty mají různou míru obecnosti. Pro efektivní využití volného času je však důležité jejich dodržování. Tyto požadavky popisují ve svých publikacích Hájek (2008, s. 76) a Pávková (2014, s. 46).

- **požadavek pedagogického ovlivňování volného času** – citlivé a nenásilné vedení a motivace k smysluplnému využití volného času.
- **požadavek dobrovolnosti** – dobrovolnost je podstatou volného času. Ačkoliv se může zdát v rozporu s předchozím požadavkem, pedagog jedince nenutí, pouze jej motivuje.
- **požadavek jednoty a specifičnosti vyučování a výchovy mimo vyučování** – určuje společný cíl. Důležitá je proto spolupráce pedagogů, jak ve výuce, tak mimo výuku.
- **požadavek aktivity** – možnost zapojení všech v daných činnostech, opět na základě dobrovolnosti. Je podporována nápaditost, samostatnost a iniciativa.
- **požadavek seberealizace** – možnost uplatnit své schopnosti a dosáhnout v něčem úspěchu. Volnočasové aktivity jsou ideální pro děti, které nedosahují úspěchů ve škole. Zažít úspěch je totiž velice důležité pro duševní vývoj.
- **požadavek pestrosti a přitažlivosti** – zaměřuje se na střídání odpočinkových a aktivizačních aktivit a rozmanitý obsah.
- **požadavek odpočinkového a rekreačního zaměření** – je potřeba zařazovat i takové činnosti, které poskytují regeneraci a odpočinek.
- **požadavek citlivosti a citovosti** – požadavek na pedagoga a jeho emocionální působení na účastníky, takovým způsobem, aby vytvářelo pozitivní zážitky.
- **požadavek orientace na sociální kontakt** – volný čas by měl být prožíván v interakci s ostatními lidmi. Je však třeba také dát pozor na dostatečné trávení času v rodinném kruhu.
- **požadavek efektivity** – organizace vybraných aktivit by měla být taková, aby se dosáhlo cílů správnými prostředky a efektivně.
- **požadavek kvality a evaluace** – souvisí s hodnocením, řízením kvality a dosažením kýžených výsledků.

2 Zájmové činnosti

Nyní se zaměříme na zájmové činnosti, jakožto samostatnou oblast výchovy ve volném čase. Vymezíme si pojem zájmové činnosti, zájmové vzdělávání a jeho organizační formy a klasifikace.

Zájmová činnost se chápe jako aktivita, která záměrně rozvíjí jedince, jeho schopnosti, zájmy a potřeby. Na rozdíl od koníčků, což jsou činnosti bez určeného cíle. U zájmových činností jde o systematické výchovné působení (Kaplánek, 2017, s 123).

Národní program rozvoje vzdělávání v ČR (Bílá kniha, 2001, s. 54) definuje zájmovou činnost jako *„souhrn výchovně vzdělávacích, poznávacích, rekreačních a dalších systematických, ale i jednorázových činností a aktivit, směřujících k účelnému a efektivnímu naplnění volného času a umožňujících získat vědomosti a dovednosti mimo organizovanou výuku. Plní funkci výchovnou, vzdělávací, kulturní, zdravotní (regenerační a relaxační), sociální a preventivní. Vede účastníky k seberealizaci a sebepoznávání, objevování vlastních schopností a jejich rozvíjení. Tím se podílí na kultivaci osobnosti, na rozvoji talentů a vede k vytváření a utužování sociálních vztahů a vazeb.“*

2.1 Zájmové vzdělávání

Zájmové vzdělávání v České republice definuje Školský zákon, konkrétně v § 111. Je zde definováno jako vzdělávání poskytující účastníkům naplnění volného času zájmovou činností se zaměřením na různé oblasti. Uskutečňuje se ve školských zařízeních pro zájmové vzdělávání, zejména ve střediscích volného času, školních družinách a školních klubech.

Zájmové vzdělávání přináší daleko více možností pro seberealizaci a objevování než klasické vyučování. Jejich velkou výhodou je to, že účastníci mají o danou činnost opravdu zájem.

2.1.1 Funkce zájmového vzdělávání

Zájmové vzdělávání plní spoustu důležitých funkcí. Bílá kniha uvádí zejména funkci výchovnou, kulturní, zdravotní, sociální a preventivní (Bílá kniha, 2001). Vymazal (1990) definuje nejdůležitější funkce takto:

- **sociálně adaptační a integrační** – spočívá v socializaci jedince a jeho začlenění do společnosti.

- **kompenzační** – doplňuje nedostatky školské soustavy, která neumí dostatečně pokrýt společenskou poptávku po vzdělání.
- **popularizační** – motivuje jedince k dalšímu vzdělávání a otevírá nové pohledy na danou oblast.
- **relaxační** – jejím cílem je dosažení uvolnění a odpočinku.
- **profylaktická** – snaží se minimalizovat ohrožení ideálního vývoje osobnosti, překonávat stereotypy osobního života a vyplnit volný čas smysluplnou a kvalitní činností.

2.1.2 Cíle zájmového vzdělávání

Hlavním cílem je zformovat samostatnou a vyrovnanou osobnost, která se zapojuje do společenských vztahů. Rozlišujeme zde cíle dovednostní (psychomotorické), poznávací (kognitivní) a hodnotové (afektivní). Hlavní myšlenkou je využití volného času co nejkvalitněji a pokud možno tvůrčím způsobem. Stejně jako u vzdělávání, je důležité si osvojovat a upevňovat vědomosti a dovednosti v co nejvyšší možné míře. Dosažením těchto znalostí bude jedinec lépe připraven na budoucí vzdělávání a uplatnění ve společnosti. Tento proces je dlouhodobý a trvá takřka celý život (Šerák, 2009; Heřmanová a Macek, 2009).

Mezi další obecné cíle můžeme zařadit:

- Motivace ke kreativnímu myšlení, logickému uvažovaní a řešení problémů.
- Vedení ke spolupráci a respektu k práci jak vlastní, tak druhých.
- Příprava k otevřené komunikaci, uplatňování svých práv a naplňování svých povinností.
- Vytváření potřeby projevovat city, prožívání životních situací, chránění fyzického a duševního zdraví.
- Vedení k toleranci a ohleduplnosti k ostatním.

2.1.3 Instituce zájmového vzdělávání

Dle ustanovení zákona č. 561/2004 Sb., o předškolním, základním, středním, vyšším odborném a jiném vzdělávání, podle § 3 vyhlášky č. 74/2005 Sb. (Česko, 2005) o zájmovém vzdělávání, patří do oblasti zájmového vzdělávání střediska volného času, školní družiny a školní kluby.

Střediska volného času, domy dětí a mládeže

Střediska volného času (dále jen SVČ) jsou školská zařízení, jejichž posláním je motivace, podpora a vedení dětí, žáků, mládeže, ale i dospělých k rozvoji osobnosti, získávání a rozvoji odborných kompetencí, zejména smysluplnému využívání volného času, a to širokou nabídkou činností v bezpečném prostředí, s profesionálním týmem pedagogů. Činnost středisek volného času se uskutečňuje ve více oblastech zájmového vzdělávání nebo se zaměřuje na konkrétní oblast zájmového vzdělávání. (MŠMT 2013-2018).

SVČ se dále dělí na Domy dětí a mládeže, které se věnují více oblastem zájmového vzdělávání a Stanice zájmových činností, které se zabývají jednou specifickou oblastí (vyhláška č. 74/2005 Sb., o zájmovém vzdělávání).

Školní družiny a kluby

Školní družiny a školní kluby jsou školská zařízení, která účastníkům poskytují trávení volného času zájmovými činnostmi navazujícími na školní výuku. Ty jsou vykonávány především ve dnech školního vyučování, zřídka i přes víkendy či školní prázdniny. Družiny i kluby většinou spadají pod školu a jejich chod řídí vedoucí vychovatelka jmenovaná ředitelem školy.

Školní družina je určena dětem přihlášeným k pravidelné denní docházce, především žákům prvního stupně. Jedna skupina by neměla přesahovat 35 dětí. Družina žákům nabízí možnost odpočinku a případně přípravy na vyučování. Poskytuje tak prostor k vyplnění času mezi vyučováním a pobytem doma.

Na družiny navazují Školní kluby, kde se hlásí žáci od 5. ročníku výše. Na rozdíl od družiny nabízí spíše spontánní aktivity odpovídající dané věkové kategorii. Nejčastěji jde o činnosti napomáhající osobnímu růstu dítěte, tedy společenskovědní, technické, přírodovědné a sportovní.

2.1.4 Formy zájmového vzdělávání

Aby bylo možné zájmové vzdělávání realizovat, je třeba jej uspořádat z organizačního hlediska, tj. koná se v určenou dobu na určeném místě. Bednaříková (2006, s. 47) charakterizuje pojem forma vzdělávání jako „vnější uspořádání vzdělávacího procesu z hlediska času, prostoru a vztahu k jednotlivým aktérům a systému vzdělávání“.

Kaplánek (2006) dělí zájmové činnosti následovně (viz Tab. 1).

Tab. 1: Rozdělení zájmových činností

Zájmová činnost		
podle obsahu		společenskovědní
		pracovně-technické
		přírodovědné
		esteticko-výchovné
		sportovní a turistické
		IT
podle formy	**podle časového rozvržení**	pravidelná
		příležitostná
	podle počtu účastníků	individuální
		skupinová
		hromadná
	podle podmínek účasti	zájmové útvary
		skupinové aktivity
		otevřená činnost

(Zdroj: Metodiky zájmových činností 1)

Klasifikace podle obsahu

Podle obsahu dělíme zájmové činnosti takto (Hájek, 2008, s. 167; Němec, 2002, s. 23):

- **společenskovědní** – vedou k poznání současného společenského dění i historie. Vytváří vztah k vlasti, demokracii a tradicím našeho národa. Poskytuje spoustu možností pro individuální přístup (např. sběratelství).
- **pracovně-technické** – pomáhají zlepšovat manuální zručnost a technické vědomosti. Pomocí těchto činností jedinec pochopí nutnost neustálého vzdělávání v době velkého rozmachu vědy a techniky.
- **přírodovědné** – vedou k prohlubování vědomostí o přírodě a navádí k pozitivnímu vztahu k ní. Okruhy jsou pěstitelství, chovatelství, pozorování živé i neživé přírody a ochrana životního prostředí.
- **esteticko-výchovné** – rozvíjí hudební, literární, výtvarný a hudebně-pohybový projev. Mají vliv na rozvoj kreativity a představivosti. Dělí se na výtvarné, hudební a literárně-dramatické.
- **sportovní a turistické** – napomáhají ke zlepšování fyzické kondice. Se zhoršující se fyzickou zdatností mládeže jsou tyto činnosti čím dál tím důležitější. Zájem o sporty bohužel stále klesá, což vyúsťuje v nezájem o pohyb i v dospělosti.
- **IT** – jde o nejnovější a nejrychleji se rozvíjející oblast, která učí počítačovým dovednostem a rozvíjí informatické myšlení.

Klasifikace podle formy

Pro nás jsou důležité hlavně pravidelné skupinové činností. Ty podle Hájka (2008, s. 169) dělíme takto:

- **kroužek** – menší zájmový útvar. Účastníci mají zájem o stejnou aktivitu.
- **soubor** – má obyčejně více členů než kroužek a jeho činnost směřuje k veřejné produkci. Nejčastěji se jedná o taneční, divadelní nebo pěvecké soubory.
- **klub** – má volnější strukturu organizace. Činnost účastníků je více pasivní (např. filmový klub)
- **oddíl** – používají ho tělovýchovné nebo turistické útvary. Případně může být součástí většího celku (např. skaut).
- **kurz** – má jasně vymezenou délku trvání (méně než rok), prezentuje tedy ucelenou vzdělávací jednotku. Jeho cílem je osvojení si vědomostí a dovedností z určité oblasti. Především se zaměřuje na obsah a dosažení cíle. Kurzy jsou nejčastěji vzdělávacího charakteru. Pro naše potřeby, tedy rozvoj informatického myšlení, je kurz ideálním útvarem.

2.2 Vzdělávání formou kurzu

Cílem kurzu je nabídnout okruh vědomostí určeného rozsahu v daném čase. Může mít charakter všeobecně nebo odborně vzdělávací. Účastníci se do kurzu přihlašují, obyčejně na konci skládají zkoušku a po úspěšném splnění získají certifikát nebo diplom. Může být prezenční, kdy jsou účastníci fyzicky přítomni v učebně nebo distanční (tzv. e-learningové), kdy účastníci dostávají studijní materiály online a pracují z domova ve volném čase. Lze jej definovat jako krátkodobou i dlouhodobou organizační formu, umožňující tvůrčí, poznávací, rekreační a výukové volnočasové aktivity. Slouží k rozvoji osobnosti, uspokojení osobních zájmů a prohloubení znalostí. Uskutečňuje se ve školských zařízeních pro zájmové vzdělávání, hlavně ve střediscích volného času a domech dětí a mládeže. Není zakotven v rámcových vzdělávacích programech a každá organizace si musí vytvořit vlastní dokumenty. Ty se odvíjí od konkrétní instituce a jejich možností. Zaměřují se na cíle, časový plán a formu a obsah aktivity. Dále obsahují způsob, jakým bude aktivita ukončena.

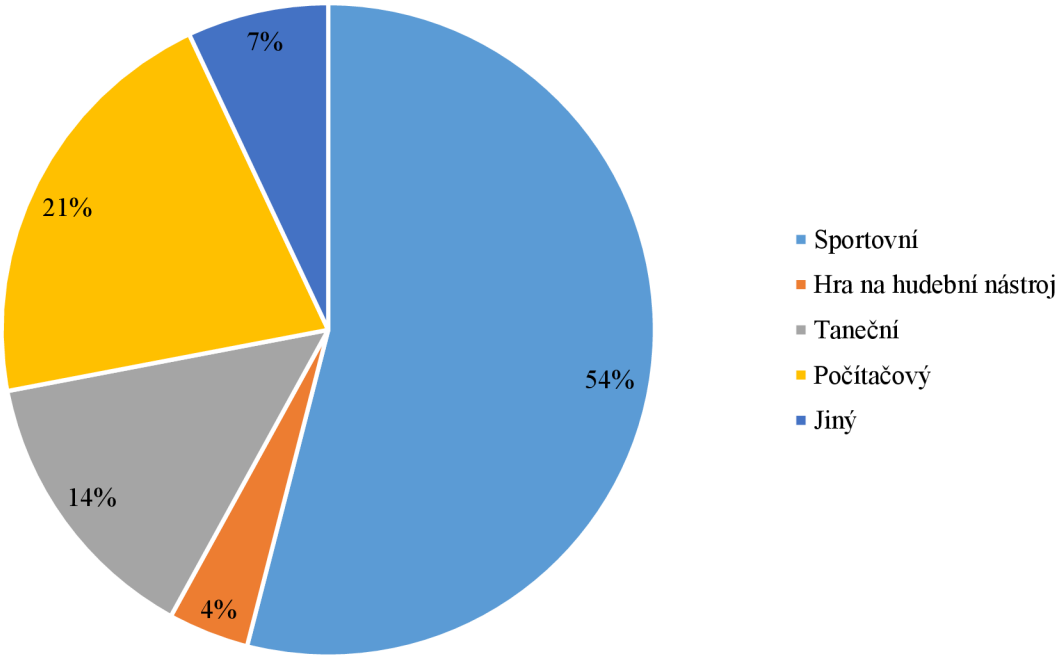
Cílem odborného vzdělání je, mimo teoretických znalostí, doručit hlavně praktické dovednosti. Obory jako např. informační technologie se velmi rychle vyvíjejí a oficiální vzdělávací dokumenty dostatečně rychle nereagují. Současně nelze jít během vzdělávacího

procesu do takové hloubky kvůli nedostatku prostoru. Právě tento problém řeší různé zájmové kurzy. Díky nim se účastníci mohou vzdělávat v oborech, které je zajímají a rozšiřovat si vědomosti. Zároveň mají profesionální vedení.

2.3 Počítačové kurzy

S rozvojem informačních technologií a postupným separováním do jednotlivých podoborů, přibývá počítačových kurzů a kroužků, které se specifikují na určité oblasti. V dnešní době už nestačí obecný počítačový kurz, snažící se obsáhnout široké pole vědomostí. Naopak je to prakticky nemožné. Ideálním se jeví, začít se základy ovládání PC a operačního systému a dále navazovat úžeji zaměřenými kurzy. Nabídka těchto kurzů je u nás již velmi pestrá a pokrývá poptávku tak, aby se každý mohl vydat směrem, který ho opravdu láká.

Dle výzkumu Šiškové (2010) na Olomouckých základních školách navštěvuje počítačový kroužek 21% z celkově dotazovaných (viz Graf 1). Po sportovním se jedná o druhý nejnavštěvovanější. A ačkoliv je výzkum z roku 2010, byl zde jasně patrný nástup elektronických technologií a jejich zájmu o ně.



Graf 1: Jaký zájmový kroužek navštěvuješ (%)

(Zdroj: Šišková, 2010)

Existují např. kurzy:

- **herní** – mimo jiné učí zdravému životnímu stylu, angličtinu, spolupráci, komunikaci v týmu, fair-play a osobnímu rozvoji. Kromě PC se využívají herní konzole jako Xbox, PlayStation či Kinect.
- **grafické** – učí práci s multimédii, animacemi, úpravu obrázků a fotek v různých programech. Na výběr je mezi 2D a 3D grafikou, rastrovou a vektorovou grafikou.
- **DIY (Do It Yourself)** – je určený pro náročné a kreativní kutily. Nejčastěji se používají stavebnice Arduino a Raspberry Pi, které dovolují vytvoření prakticky libovolného projektu propojeného s reálným světem. Velký zájem je rovněž o 3D tisk**.**
- **virtuální (VR) a rozšířené (AR) reality –** moderní oblast IT, která využívá nejnovějších technologií a má do budoucna velký potenciál.
- **robotiky a kybernetiky –** v současnosti nejpopulárnější jsou stavebnice robotů Lego. Kurzy spojují více dovedností dohromady: manuální zručnost, kreativitu a programování.
- **elektroniky –** programování mikroprocesorů, průmyslových robotů a PLC; zapojování jednoduchých obvodů a součástek.
- **počítačové bezpečnosti –** učí jak se bezpečně chovat v prostředí internetu, ubránit se neoprávněným útokům a virům, bezpečně komunikovat a přenášet data, zálohovat apod.
- **psaní všemi deseti –** velmi užitečná dovednost, kterou využije jak programátor, tak běžný uživatel počítače. Se zvyšujícím se množstvím práce, kterou vykonáváme pomocí PC, dokáže tato dovednost výrazně zkrátit čas strávený u PC.
- **programování** – u mladších účastníků jsou vyučovány programovací jazyky jako Scratch nebo Baltík, pokročilejší se mohou pustit do plnohodnotných jazyků, jako jsou Python, C# nebo Java.

Ruku v ruce se všemi uvedenými typy kurzů jde rozvoj informatického myšlení obecně.

3 Informatické myšlení

Zrychlující se vývoj technologií přinesl mnoho radikálních změn do všech aspektů života a nepochybně ovlivnil fungování naší společnosti v posledních desetiletích. Technologické inovace vedoucí k modernizaci průmyslu, obchodu a domácností vedly k velkému počtu nových konceptů souvisejících s digitálními a informačními technologiemi a jejich využitím. Jedním z nich bylo informatické myšlení (dále IM), zavedené Jeannette Wing v roce 2006, jako nevyhnutelná dovednost moderního člověka, který dokáže plně využívat výhod digitálních technologií a počítačových algoritmů k řešení každodenních problémů.

Odpověď na otázku, co je IM, je ve skutečnosti docela komplikovaná. Zastánci IM až donedávna trávili spoustu času debatami, jak jej definovat. V roce 2011 ve Washingtonu byl uspořádán workshop, kde se sešla spousta individualit ze světových IT firem, aby posoudili, jaká by měla být podstata IM. Někteří na tomto semináři argumentovali ve prospěch přesné definice. Jiní naopak tvrdili, že snaha o striktní definici je zbytečná (Voogt a kol., 2015).

Podle druhého názoru by pochopení IM nemělo být provedeno přijetím definice v obvyklém smyslu (vytvořením seznamu podmínek, které musí definice splňovat). Voogt a kol. tvrdí, že definovat IM je podobně obtížné, jako definovat, co je to „hra“. Musí v každé hře stát alespoň jeden hráč proti druhému? Má každá hra koncept vítězství? Měla by každá hra obsahovat nějaký prvek štěstí nebo náhody? Stejně jako naše chápání hry, přístup k definování IM by měl být jemnější a vnímán jako řada podobností a vztahů, které se protínají a překrývají.

Existují další důvody, proč je definice IM tak náročná. Zaprvé, IM pevně souvisí s informatikou, jejíž definování může být samo o sobě problematické. Stejně jako IM zahrnuje i informatika řadu abstraktních i konkrétních myšlenek. Oba pojmy sdílejí univerzální použitelnost a ta také činí IM obtížně výstižně definovatelným.

IM je také myšlenkou, která je zároveň nová i stará. Nová v tom smyslu, že se náhle stala horlivě diskutovaným tématem po přednášce Wingové (Wing, 2006). Mnoho klíčových myšlenek však bylo diskutováno již několik desetiletí zpátky a lidé je různým způsobem interpretovali. Například již v roce 1980 Seymour Papert z Massachusetts Institute of Technology propagoval koncept, který nazval „procedurální myšlení“ (Papert, 1980). Sdílel mnoho myšlenek s tím, jak nyní přemýšlíme o IM. Papert se zaměřil na to, aby studentům poskytl metodu řešení problémů, s využitím počítačů jako nástrojů, pomocí procedurálního myšlení. Hlavní myšlenkou bylo, že se studenti naučí vytvářet algoritmická řešení, která by pak mohl provádět počítač; k tomu použil programovací jazyk Logo. Papertova práce inspirovala vznik IM, ačkoliv IM se od původní myšlenky v některých ohledech liší.

Podle Wingové je IM myšlenkovým procesem, který umožňuje formulovat problém a popsat tak jeho řešení způsobem, který lze efektivně zpracovat počítačem, strojem nebo dokonce člověkem (Wing, 2014). Dá se tedy říci, že obecně se jedná o způsob řešení problému, který se zaměřuje na jeho popis, analýzu a nalezení účinného řešení, s důrazem na systematický přístup a využití konceptů známých v oblasti informatiky. Od prvního představení konceptu IM proběhlo velké množství mezinárodních diskuzí o přesné definici a integraci IM do učebních osnov vzdělávacích systémů v podstatě celého světa. Představení konceptu IM v akademických kruzích podpořilo pedagogickou rozpravu o roli digitálních technologií ve vzdělávání a zavádění informatiky a programování do osnov, existujících téměř od roku 2000 (Tran, 2017, Klement, 2018).

Mnoho zemí si, s ohledem na zrychlující technologický vývoj, uvědomuje, že výuka zaměřená pouze na výchovu aktivních uživatelů je nedostatečná. Mimo jiné k tomu přispívá fenomén tzv. digitálních domorodců (digital native), který Prensky popsal již v roce 2001. Dnešní žáci a studenti jsou součástí generace digitálních domorodců, která od dětství vyrůstá s digitálními technologiemi a je pro ně přirozené, že mohou svá elektronická zařízení používat každý den od útlého věku (Prensky, 2001). Jak technologie postupují, stupeň jejich využívání se v populaci zvyšuje a školský systém musí na tyto změny pružně reagovat.

Obecně lze říci, že IM přináší novou perspektivu v problematice vývoje digitálních a informačních technologií a nových kompetencí, které by měli žáci získat v rámci základního vzdělání. Navzdory skutečnosti, že dnešní generace studentů tráví velkou část svého života používáním digitálních technologií, dávají přednost digitální zábavě a sociálním sítím (Klement, 2018). Podstatou revizí ICT osnov je právě tendence přizpůsobit školský systém formálnímu rozvoji již existujících digitálních a informačních dovedností žáků v rámci základního vzdělávání.

Obsah předmětů dříve primárně zaměřených na obsluhu počítače a rozvoj digitální gramotnosti u žáků se dramaticky mění, a je zde tendence inovovat učební osnovy tak, aby jejich klíčová složka již nespočívala čistě v digitální gramotnosti, ale také v algoritmizaci, programování a dalších počítačových dovednostech. Cílem těchto revizí je: dosáhnout výuky, která nevede k pouhému využívání digitálních technologií na pasivní uživatelské úrovni, ale k rozvoji dovedností a kompetencí, které umožní žákům plně využít potenciál moderních technologií při řešení komplexních problémů.

Ačkoliv práce s počítačem a cílený rozvoj digitálních a komunikačních dovedností mají stále značný význam, existuje tendence přesunout cílený rozvoj těchto dovedností do mezioborové oblasti v rámci modernizace celého vzdělávacího sektoru (Balanskat, 2018).

Epidemická situace kolem COVID-19 ukázala, že v moderním světě není nutné se bát používání digitálních technologií ve výuce mimo hodiny výpočetní techniky, ale že digitalizace vzdělávání je naopak nevyhnutelná. V současnosti by měla být výuka digitální gramotnosti proveditelná prakticky napříč předměty.

Některé vyspělé evropské země dlouhodobě pracují na programech, které aplikují digitální gramotnost nad rámce ICT hodin. Švédsko, jedna z nejvyspělejších zemí v této oblasti, plně implementovalo svou reformu zaměřenou na modernizaci vzdělávání v roce 2018, která vyžaduje, aby učitelé poskytovali žákům dané studijní materiály, pomocí kterých můžou studovat digitální technologie a dále se rozvíjet.

Tato strategie hovoří o zcela nové klíčové kompetenci učitele, jmenovitě schopnost organizovat a sdílet svou práci pomocí digitálních médií. Očekává se, že efektivní používání digitálních technologií ve výuce přispěje k plné digitalizaci švédského vzdělávání v budoucnosti (Balanskat, 2018). V budoucnu budou v tomto trendu pravděpodobně pokračovat všechny technologicky vyspělé země.

Posunutí výuky digitálních technologií do mezioborového spektra poskytuje prostor pro kurikulární reformu zaměřenou na cílený rozvoj IM mezi žáky, kteří to budou v budoucnu potřebovat jak v běžném životě, tak při výběru kariéry. Porozumění technologiím, ne-li konkrétní programovací schopnosti se stávají nezbytným předpokladem každého člověka v moderní informační společnosti při úspěšném zapojení do trhu práce. Milníkem v této diskusi byla studie z roku 2012 „Shut Down or Restart“ Britské Královské společnosti, která odůvodnila potřebu reformy osnov pro výuku informatiky potřebou rovného přístupu ke vzdělání v IT. Podle studie má každé dítě právo na základní vzdělání v oboru informatiky za účelem získání potřebných předpokladů pro další studium a získání kvalifikace v oboru (The Royal Society, 2012).

V mnoha vyspělých zemích vedla potřeba odstranit překážky v přístupu ke kvalitnímu informatickému vzdělání k nové koncepci výuky zahrnující rozvoj informačních a digitálních kompetencí. Což nicméně není žádný nový trend. Ještě před první integrací koncepce rozvoje IM do osnov byla tendence provádět kurikulární revize ke zlepšení výuky informatiky a programování.

Vyspělé země, jako jsou USA, Švédsko a Japonsko, přijaly nové standardy pro obsah učebních osnov, rozšířily je o algoritmizaci a programování zaměřené na profesní rozvoj učitelů a propagaci informatiky studentům a veřejnosti (Tran, 2017). Největší přínos těchto reforem se očekává ve včasném seznámení žáků s programováním, což jim později pomůže přizpůsobit se složitějším aspektům informatiky a komplexním programovacím jazykům (Evropská komise,

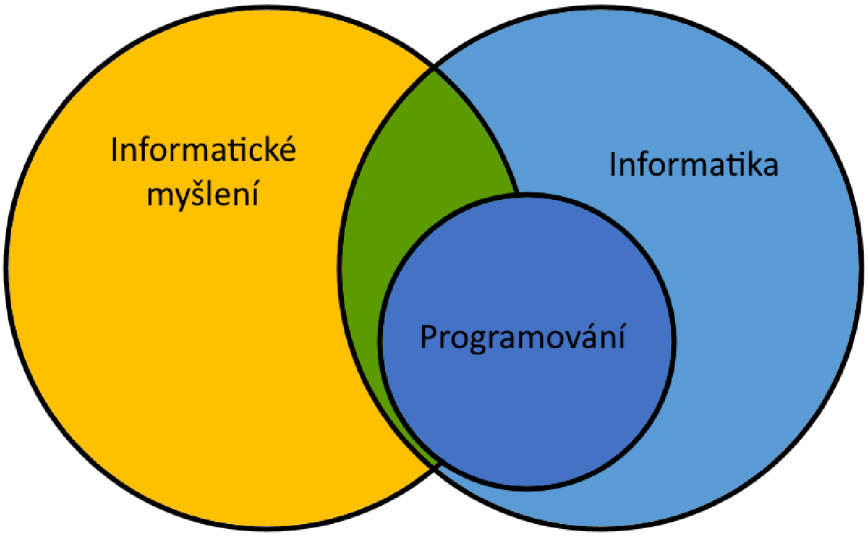
2013). Z tohoto pohledu je rozvoj IM důležitou podmínkou, nezbytnou pro efektivní a kreativní výuku technologií, programování a další rozvoj studenta nejen v oblasti informatiky.

3.1 Definice pojmu informatické myšlení

Přesná definice IM je předmětem diskuzí, které se konají prakticky od prvního uvedení koncepce v roce 2006. Doposud však nebylo možné najít uspokojivou definici přijatelnou pro všechny strany akademické debaty. Neshody v samotném pochopení konceptu silně ovlivňuje výslednou implementaci národních učebních osnov, jakož i jejich obsahu a provádění revize.

V praxi můžeme najít dva přístupy k definování IM. První spočívá v obecné charakteristice vlastností, dovedností a předpokladů týkajících se použití IM při řešení problémové situace. Tato definice je obvykle příliš všeobecná a popisuje IM jako kognitivní myšlenkový proces a v mnoha případech je výchozím bodem pro další konkretizaci a konceptualizaci. Ve své nejzákladnější podobě je IM schopnost „myslet jako programátor" (Wing, 2014). Všeobecná definice vidí IM jako porozumění světu silně pozměněnému technologiemi a zdůrazňuje potenciál řešit složité problémy v široké škále oblastí, často bez použití samotného programování (Wing, 2006). Tedy jde spíše o schopnost řešit problém obecně, než skutečná schopnost programovat.

IM je často vnímáno jako interdisciplinární koncept nad rámec samotné informatiky. Jedná se o soubor dovedností a kognitivních nástrojů potřebných k řešení úkolů, navrhování systémů, porozumění chování, automatizace procesů a řešení problémů pomocí počítače (NAS, 2011). Jedná se spíše o kognitivní koncept, než specializovanou vědeckou disciplínu. Zároveň, jako bylo řečeno výše, ve své základní povaze nemusí být IM nutně spojeno s programovacím jazykem (Wing, 2014) nebo výpočetními technologiemi. Nejde však ani o úplné oddělení IM od programování. Programování je v oblasti IM nezbytné, někteří autoři však tvrdí, že není nutné používat formální programovací jazyk. IM může být použito při řešení problémů, které nevyžadují použití programovacího jazyka, a to jak v praxi tak při výuce. Na základě těchto úvah lze říci, že IM zasahuje do informatiky a využívá své znalosti a základní principy a zároveň může používat programování jako nástroj. Není na něm však závislé. Tento vztah je reprezentován grafem konceptu IM (viz Obr. 1).



Obr. 1: Ilustrace konceptu IM

(Zdroj: KLEMENT, Milan, Tomáš DRAGON a Lucie BRYNDOVÁ, 2020)

Jedná se o přístup, který přímo neuvádí do souvislosti IM a schopnost používat určitý programovací jazyk. Způsobuje rozkol na akademické legislativní a pedagogické úrovni. Některé národní osnovy se ztotožnily s konceptem založeným na formálním programovacím jazyku, zatímco jiné zmiňují potřebu programování (Balanskat, 2018) přímo v základní definici IM.

Druhý způsob je, definovat konkrétní složky a dílčí koncepty, které jsou s ním spojeny. Hlavní motivací pro tuto konkretizaci je neurčitost původního konceptu, který není vhodný pro pedagogickou a výukovou aplikaci. Pokusy definovat konkrétní složky IM vychází z potřeby stanovit didaktické cíle, které jsou měřitelné, dosažitelné, přijatelné, realistické a vázané na termín, což je zásadní pro kvalitní aplikovanou výuku.

V současné době existuje velké množství přístupů k definování dílčích komponent a konceptů, od podrobné analýzy osnov, obsahů výuky a očekávaných výstupů, cíleného rozvoje schopností, dovedností a vlastností žáků, k počítačovým konceptům spojeným s konkrétním programovacím prostředím nebo vědeckým oborem (NAS, 2010, Tran, 2017). Budeme se zabývat především nejpoužívanější definicí IM v kontextu státní legislativy, což je dokument vydaný CSTA (Computer Science Teachers Association) a ISTE (International Society for Technology in Education) v roce 2011.

Dokument nazvaný Operational Definition of Computational Thinking for K-12 Education, který se stal základem pro většinu evropských kurikulárních dokumentů, uvádí šest schopností a dovedností, které tvoří základ IM a je rozšířen o pět charakteristik, které jsou relevantní pro koncept IM. Obvykle jsou tyto charakteristiky IM znázorňovány schematicky

jako schopnost algoritmizovat, rozložit problém, generalizovat, evaluovat a abstrahovat. Někteří autoři tyto charakteristiky kombinují, například mnoho autorů považuje automatizaci a algoritmizaci za identickou.

Ačkoliv definice a přístupy k IM se často velmi liší autor od autora, většina se shoduje na této zjednodušené definici. I přes částečné variace lze říci, že IM je obecně chápáno jako způsob myšlení, jak řešit problémy metodicky s využitím principů známých z informačních technologií. Jde tedy o kognitivní proces, který je nezbytný pro efektivní řešení složitého problému a vyjádření procesu tak, aby byl strojově zpracovatelný. Řeč je o automatizaci, algoritmizaci, reprezentaci, ladění, modelování a strukturování. V závislosti na vnímání může být IM velice úzce spojeno s programováním nebo chápáno jako způsob myšlení, který je dokonce použitelný bez přímého spojení s formálními programovacími jazyky.

3.2 Subdomény informatického myšlení

Od začátku mezinárodní debaty o integraci rozvoje IM do výuky se objevují pokusy definovat konkrétní subdomény IM. Primárním cílem tohoto procesu je: konkretizovat jinak velmi obecnou definici fenoménu IM, která není vhodná pro praktickou implementaci do školského systému. V současné době se většina národních kurikulárních definic zakládá na nebo se obecně shoduje s definicí charakteristik a schopností podle CSTA a ISTE z roku 2011.

Dokument CSTA a ISTE z roku 2006 identifikuje šest oblastí, které zahrnuje proces IM a dalších šest dovedností a kompetencí. Jedná se tedy o definici založenou na dovednostech, kompetencích a přístupech. Konkrétně to jsou schopnosti formulace řešení, zpracování dat, reprezentace dat pomocí abstrakce, jako jsou modely a simulace, automatizace a algoritmizace, nalezení optimálního řešení a schopnost generalizovat a aplikovat proces k řešení problému. Tento dokument byl aktualizován o pět let později, aby zdůraznil prvky abstrakce, automatizace a analýzy (CSTA a ISTE 2011).

Tyto definice byly později zjednodušeny mnoha autory a redukovány na základní prvky, které popisují podstatu původních definic. Následující srovnávací tabulka (viz Tab. 2) uvádí seznam dílčích komponent IM založený na definicích CSTA a ISTE a klíčová slova a fráze použité v této definici podle Chen (2017), na základě kterých definujeme odpovídající dovednosti, spojené s těmito koncepty.

Tab. 2: Definice oblastí pro rozvoj informatického myšlení

Původní definice CSTA a ISTE	Klíčová slova	Odpovídající IM dovednost
Formulovat problémy pro strojní řešení	Formulace	Syntaxe, programování
Logicky organizovat a analyzovat data	Data	Zpracování dat
Reprezentace dat pomocí abstrakce	Reprezentace	Modelování
Automatizace řešení pomocí algoritmizace	Algoritmizace	Algoritmizace, automatizace
Analýza možných řešení pro dosažení nejefektivnější kombinace	Nejefektivnější kombinace	Abstrakce, optimalizace
Generalizace a použití konkrétního procesu k řešení problému	Generalizace	Evaluace, ladění, generalizace

(Zdroj: KLEMENT, Milan, Tomáš DRAGON a Lucie BRYNDOVÁ, 2020)

Přestože přístupy k samotnému IM se stále velmi liší, v současné době se dle dostupné literatury a legislativních dokumentů ukazuje, že většina navrhovaných dílčích složek se přesně shoduje s algoritmizací, abstrakcí, laděním nebo evaluací, rozkladem a generalizací (Bocconi a kol., 2016). Možné alternativy k těmto oblastem jsou založeny na konkrétních národních koncepcích IM.

Je třeba poznamenat, že ačkoliv mnoho autorů, včetně Wingové a CSTA a ISTE, považuje automatizaci jako klíčový komponent IM, diskutuje se, zda to není jen součást algoritmického myšlení (Angeli, 2020). Není jasné, jak přesně oddělit automatizaci od algoritmizace v pedagogické praxi. Dokonce ani v učebních osnovách nejsou tyto oblasti obecně definovány jako samostatné koncepty. Podobně oblasti syntaxe a zpracování dat, přestože jsou výslovně zmíněny v původním dokumentu CSTA a ISTE jsou z hlediska učebních osnov pokryty nekonzistentně.

Tento trend je způsoben nejasnou definicí programování v konceptu IM a pokusem oddělit je od digitální gramotnosti. Autoři, kteří přistupují k IM v širším smyslu, nespojují IM s použitím programovacího jazyka a obecně ani se znalostí formální syntaxe. Podobně kurikulární dokumenty, které zahrnují schopnost pracovat s daty a informacemi v oblasti digitální gramotnosti, nemusí tuto oblast zahrnovat do učebních osnov týkajících se IM. Zde je uvedeno pouze pět základních oblastí IM, které obecně popisují definice IM a které se nejčastěji nacházejí v dokumentech pedagogické, didaktické a legislativní povahy.

- **Abstrakce** – je považována za nejdůležitější součást IM (Wing, 2014). V kontextu IM je to schopnost zjednodušit problém do jeho základní podoby, aby se nevytratily podstatné informace a poté problém schematicky znázornit. Pracuje s definicí vzorců, generalizací, reprezentací, simulací, implementací, parametrizací, optimalizací a dalšími dovednostmi potřebnými k úspěšném řešení složitých problémů.

- **Algoritmizace** – schopnost a dovednost najít a adekvátně formulovat efektivní řešení konkrétního problému. Diskutuje se, zda to v kontextu IM znamená, formulovat řešení ve formálním programovacím jazyce, nebo jde o koncept, který je na praktickém programování zcela nezávislý.
- **Dekompozice** – rozložení problému, spojené s modularizací. Schopnost rozdělit celek na dílčí části a dále s nimi pracovat. Úzce navazuje na abstrakci, protože se jedná o řešení částí problému.
- **Systematická evaluace –** zahrnuje pojmy jako analýza, ladění a parsování. Umožňuje výsledek situace a fungování algoritmu předvídat na základě kritické analýzy. Pracuje s testováním, analytickým a logickým myšlením a hodnocením.
- **Generalizace –** znamená identifikaci podobností a souvislostí problémů, která by měla vyústit v návrh univerzálního řešení, aplikovatelného na více situací. Jde o schopnost řešit více problémů na základě podobnosti a využití učení.

S ohledem na relativně dlouhodobou diskuzi o definici IM a jeho součástí, je pochopitelné, že dokument CSTA a ISTE není jedinou dostupnou charakteristikou tohoto konceptu. Někteří autoři, zejména v kontextu praktického testování a metodiky výuky rozvoje IM, inklinují ke starším konceptům, které nabízí jasnější standardy hodnocení. Populárním pojmem v tomto ohledu je tzv. 3D framework, který v roce 2012 popsali Brennanová a Resnick pod záštitou MIT.

3D framework představuje poněkud odlišný přístup ke specifikaci IM než CSTA a ISTE. Brennanová a Resnick vyvinuli systém aplikovatelný na výuku žáků základní školy využívající specifické programovací prostředí s názvem Scratch a jejich definice komponent IM je tedy plně spojena s použitím programování. 3D framework představuje tři hlavní dimenze aplikovaného IM, které jsou doplněny podrobnými koncepty informatiky a programování, které by měl informaticky smýšlející jedinec být schopen použít.

První dimenze IM podle tohoto modelu pokrývá koncepty IM, které zahrnují schopnost používat sekvence, smyčky, události, paralelní programování, podmínky, operátory a zpracování dat. Druhá dimenze zahrnuje procvičování IM, což znamená, že určitě postupy musí žáci nutně procvičovat, aby se jejich IM mohlo efektivně rozvíjet. Zejména to znamená inkrementační a iterační schopnosti ve světle vývoje programu, testování a ladění, univerzálnost, abstrakce a modularizace. Třetí dimenze jsou tzv. perspektivy IM, tedy reflexe, sebevyjádření, komunikace, spolupráce a přenos znalostí (Brennanová a Resnick, 2012).

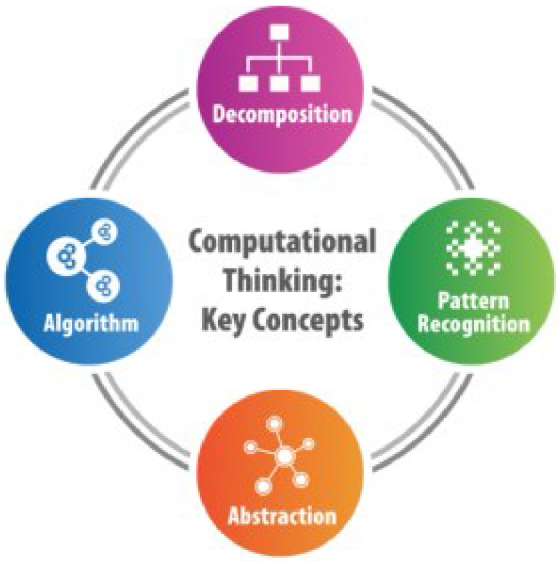
I když se tento koncept prakticky nepoužívá jako výchozí bod pro legislativní implementaci IM do školského systému, z důvodu jeho aplikovaného základu je považován za výhodný při hodnocení výkonu žáků při rozvoji IM ve školách.

3.3 Podmínky informatického myšlení

Obsah této kapitoly shrnuje zásadní podmínky IM, kterými jsou:

- pohled na řešený problém z různé perspektivy
- rozklad komplexního problému na více menších pod-problémů
- schopnost abstrakce
- dokončení řešení problému v konečném čase
- schopnost komunikovat informace tak, aby byly srozumitelné
- formulace problému tak, aby bylo možné využít počítač k jeho řešení
- logická organizace a analýza dat

Postup řešení problému je názorně předveden na obrázku č. 2.



Obr. 2: Podmínky informatického myšlení

(Zdroj: *Computational Thinking in K-12 Education*. 2017. Dostupný z: https://cspathshala.org/2017/10/25/computational-thinking-curriculum/)

Při čtení jednotlivých částí obrázku budeme postupovat po směru hodinových ručiček. Oblast dekompozice (Decomposition) představuje snahu o rozložení problému na menší pod-problémy, jejichž jednotlivým řešením se dostaneme k řešení finálnímu.

Oblast Pattern Recognition lze přeložit jako hledání vzorů, tedy opakujících se kroků řešení (algoritmů). Jedná se o důležitou výhodu objektově orientovaného programování (dále OOP). A tou je znovupoužitelnost kódu. V IM se však nejedná pouze o zdrojový kód, ale o jakýkoliv sled kroků.

Oblast algoritmizace je esenciální součástí IM. Jednotlivé kroky definované v předchozím odstavci je potřeba seřadit tak, aby jejich provedení odpovídalo správnému postupu. Pro vizualizaci správného sestavení algoritmu pomůže vývojový diagram, v kterém lze jednoduše zjistit, jaký prvek je chybový.

Poslední oblast abstrakce, kterou zmiňuje Wingová, prakticky znamená odstranění těch prvků, které jsou pro naše řešení irelevantní. Je důležitá pro zaměření se na podstatu problému.

3.4 Rozvoj informatického myšlení ve výuce

Ačkoliv akademická debata dosud nedosáhla úspěšné definice IM způsobem, který by byl uspokojivý pro všechny, vyspělé země se dohodly na implementaci rozvoje IM do národních osnov. Podstatou integrace do vzdělávacího systému je revize školních osnov s cílem připravit žáky na moderní informační společnost, rozvíjet jejich schopnost využívat moderní technologie a usnadnit jejich uplatnění na trhu práce (Klement, 2018).

V praxi však implementace IM do kurikulárních systémů čelí mnoha komplikacím. Kromě problémů s legislativou a definicí samotného pojmu „informatické myšlení“ existují další důležité problémy na mikro i makro úrovních školského systému.

Dnes je velká většina zemí stále v procesu implementace IM do školních osnov a pro mnoho z nich je to velmi obtížná, téměř radikální změna. Obzvláště problematické je provedení revize v těch zemích, které nemají zavedenou tradici výuky programování na školách, a proto musí revidovat prakticky všechny oblasti řízení vzdělávání. Praktická integrace IM do školních osnov obecně závisí na jednotlivých školách, jejich technologickém a ekonomickém zázemí, zdrojích, dostupném čase, kvalifikaci jednotlivých učitelů a jiných okolnostech. Tyto okolnosti často znesnadňují implementaci, protože školy nemají dostatečnou podporu k usnadnění revize vzdělávání.

V Evropě byla do mapování problémů s implementací IM ve školách zapojena European Schoolnet. Podle výsledků z tohoto průzkumu byla nejvýznamnějším problémem právě nedostatečná kvalifikace učitelů, zejména pokud jde o učitele nižšího stupně (Balanskat, 2018). I když rozvoj IM nemusí nutně souviset s výukou informatiky a většina autorů doporučuje integraci IM do předmětů jako výtvarná výchova, hudební výchova nebo

matematika (Bocconi, 2016), je realizace takové výuky v současné době v mnoha zemích nemožná, právě kvůli nedostatku financí, kvalifikace podpory pro učitele.

Prakticky všechny země, které podnikly kroky k integraci IM do svých osnov v posledním desetiletí, poukazují na nedostatečnou podporu učitelů. Zároveň zkušenosti z řady zemí, které již IM implementovaly do svého národního vzdělávání, ukazují, že důležitým aspektem při jakékoliv formě revize osnov informatiky je příprava podrobných materiálů pro učitele.

Kromě ekonomických a organizačních aspektů je hlavní problém ve zvýšených požadavcích na dovednosti a kompetence učitelů, které přináší integrace programování do škol. Bez ohledu na globální koncept a definici termínu „informatické myšlení“ je považováno za nejefektivnější nástroj pro rozvoj IM právě programování.

Programování hraje klíčovou roli při praktické implementaci IM ve výuce. Podle studie analyzující plánovanou kurikulární revizi oboru informatiky a výpočetní techniky, třináct zkoumaných evropských zemí se zaměřilo především na rozvoj dovedností žáků v oblasti řešení problémů nebo kritického a logického myšlení. Sedm zemí se zaměřilo hlavně na zavedení programování. Dalších sedm zemí pouze rozšířilo již existující koncepci vzdělávání (Bocconi a kol. 2016).

V současné době existuje mnoho studií potvrzujících pozitivní dopad výuky programování od raného dětství. Kognitivní a matematické dovednosti, dovednosti řešení problémů, soustředění a sebevědomí se zlepšily u žáků, kteří se na základních školách účastnili kurzů programování (Tran, 2017). Cílený rozvoj IM může přispět k rozvoji kompetencí jako např. čtení a psaní (Wing, 2006). Zároveň integrace programování do povinného vzdělávání na nižším stupni pomáhá eliminovat genderové stereotypy s ohledem na počítačové dovednosti.

Mezi profesionálními programátory stále přetrvává výrazný rozdíl mezi pohlavími. Ačkoliv je tento trend znám po celém světě, není úplně jasné, proč zaměstnání v IT vyhledává více mužů než žen. Mnoho průzkumů však zmiňuje u žen problémy se sebevědomím. Podobné trendy se také objevují mezi žáky základních škol.

Většina průzkumů nicméně potvrzuje, že předpoklady pro programování a učení se informatiky se mezi chlapci a dívkami neliší. Jediný rozdíl je právě v přístupu a motivaci k programování, kde mají dívky tendenci podceňovat své schopnosti. Budování pozitivního přístupu v oblasti informatiky u žáků raného věku by mohlo tyto tendence eliminovat.

Široká integrace rozvoje IM a programování do základních osnov, zejména pokud tato reforma zahrnuje žáky prvního stupně, může vést k odstranění překážek ve formě genderových předsudků. Mnoho zemí kombinuje implementaci IM do učebních osnov s popularizací

informatiky a připraveností všech absolventů základního vzdělání využívat nové technologie jak v osobním tak pracovním životě. Národní kurikulum zajišťuje rovnost vzdělání pro všechny občany na území státu. Proto je možné, že zahrnutí IM do učebních osnov povede k rovnováze mezi pohlavími na trhu práce v oblasti IT.

3.5 Podpůrné materiály pro rozvoj informatického myšlení

Spolu s rostoucím zájmem států o implementaci IM do základního vzdělávání, roste poptávka po odpovídajících učebních nástrojích a materiální podpoře. Nicméně podpora pro učitele v této oblasti je v mnoha zemích nedostatečná. Významným problémem je mimo jiné nedostatek průzkumů zaměřených na jednotlivé pomůcky (Tran, 2017). Řešením by mohla být strategie na podporu spolupráce mezi učiteli, kteří učí programování (Balanskat, 2018) a řízená distribuce učebních pomůcek a materiálů. Ale takový koncept by byl časově poměrně náročný a nákladný.

Příslušné učební dokumenty musí být dodržovány při výuce zaměřené na rozvoj IM. U většiny zemí je národní učební plán závazným dokumentem. Při navrhování jakéhokoliv obsahu výuky musí být dodržovány školní osnovy a očekávané výsledky žáků. V praxi je však řada zemí nakloněna určité svobodě výuky, kde si školy individuálně vyberou vlastní způsob výuky rozvoje IM založený na jejich vlastních kapacitách.

Musíme vzít v úvahu věk žáka při výběru příslušného propedeutického programovacího jazyka. Důležitá je přehlednost vývojového prostředí a srozumitelnost syntaxe jazyka. Zároveň je však nutné zvolit takové prostředí, které nabízí dostatek prostoru pro další rozvoj a snadný přechod na složitější programovací jazyk.

Spousta učitelů, kteří učí programování na nižších stupních základních škol, mají tendenci upřednostňovat aplikace, které lze použít na tabletech místo těch, která vyžadují myš a klávesnici. Práce s tabletem nebo jiným dotykovým zařízením pomáhá rozvíjet motoriku žáka a nevyžaduje velkou počáteční koordinaci. Robotika je jedním z nejslibnějších vzdělávacích nástrojů pro rozvoj IM a zároveň velmi chválený předmět modernizace výuky na základních školách.

Pro školy, které nemají možnost využívat vzdělávací robotiku, existují levné nebo dokonce bezplatné alternativní online nástroje a metodiky pro rozvoj IM a programovacích dovedností. Tyto nástroje si představíme v následující kapitole.

4 Existující nástroje vhodné pro rozvoj informatického myšlení

Jak bylo popsáno v předchozí kapitole, tematické celky zaměřené na rozvoj IM na mnoha školách chybí, proto lze očekávat, že úroveň znalostí žáků a učitelů v této oblasti je poměrně nízká. Aby bylo možné naplnit cíle a vize digitálního vzdělávání, je nutné zaměřit se nejen na žáky samotné, ale také na skupinu učitelů informatických předmětů a poskytnout jim potřebnou podporu v podobě metodických materiálů, různých školení a kurzů zaměřených na algoritmizaci. Tato podpora by jim následně měla pomoct zajistit komplexní výuku.

Jedním z možných způsobů, jak tohoto cíle dosáhnout, je e-learning s využitím webových a mobilních aplikací (Dragon, 2019; Dragon a Klement, 2020). Vzdělávání může být poskytováno nejen v prostorách školy, ale také doma, v různých vzdělávacích zařízeních, dokonce v dopravních prostředcích apod. Některé aplikace ani nevyžadují připojení k internetu, tudíž nejsme vázáni k jednomu místu. Mnohé z těchto aplikací otevírají nové možnosti v oblasti vzdělávání, rychlosti učení a získávání znalostí a dovedností.

I samostudium žáků a učitelů může být zásadním prvkem v procesu vzdělávání, realizaci výuky a zavádění nových změn.

Existuje spousta placených, ale i volně dostupných, vysoce kvalitních zdrojů, které lze využít k rozvoji IM v oblasti algoritmizace a programování. Vybrané webové a mobilní aplikace nabízejí širokou škálu různě tematicky zaměřených kurzů, které uživateli umožňují naučit se základní i pokročilejší techniky programování v některých z oblíbených programovacích jazyků, jako jsou Python, JavaScript atd. Tyto aplikace nabízejí také mnoho dalších užitečných a zajímavých funkcí.

V této kapitole si představíme několik příkladů webových a mobilních aplikací, které lze využít při rozvoji IM v oblasti algoritmizace a programování. Aby bylo možné lépe analyzovat jednotlivé aplikace, ohodnotíme je pěti subjektivními kritérii. Tato kritéria mohou být pro uživatele také rozhodujícím faktorem při výběru vhodné aplikace.

Prvním kritériem je **dostupnost**. Zaměříme se na podmínky a zařízení, která umožňují používání vybraných aplikací. Zda jsou kurzy a další funkce přístupné pomocí webového prohlížeče, zda jsou k dispozici jako specializované mobilní aplikace ke stažení z App Store nebo Google Play, nebo kombinace obou uvedených možností.

Dalším kritériem je **obsah**. Bude nás zajímat nabídka kurzů v oblasti algoritmizace a programování, stejně jako specifické programovací jazyky, které se může uživatel naučit prostřednictvím vybrané aplikace, ale také procvičit, pokud již má nějaké předchozí znalosti.

Třetím kritériem je **úroveň interaktivity**. To znamená, jaká je míra interakce uživatele s jednotlivými prvky vybrané aplikace.

- **nízká úroveň** – uživatel otevírá interní nebo externí zdroje pomocí hypertextových odkazů; ověřuje své znalosti pomocí testů, ve kterých vybírá správné odpovědi, nebo musí vypracovat úkol mimo prostředí samotné aplikace; aplikace také může obsahovat výuková videa.
- **střední úroveň** – uživatel zadává části kódu do určených polí, přesouvá kód do správného pořadí pomocí techniky „drag and drop“; může navíc obsahovat prvky z nižší úrovně.
- **vysoká úroveň** – uživatel používá vývojovou konzoli integrovanou v prostředí aplikace; aplikace taky může osahovat nějakou formu interního fóra nebo chatu, kde může uživatel v reálném čase komunikovat s jinými uživateli a spolupracovat v rámci samostudia; může navíc obsahovat prvky z nižších úrovní.

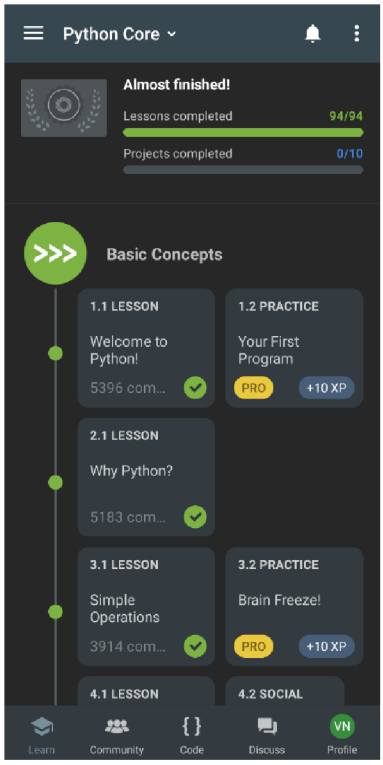
Čtvrtým kritériem je **jazyková lokalizace**, která může mít zásadní význam zejména v případech, kdy aplikace není lokalizovaná v mateřském jazyce uživatele. Cizí jazyk může představovat problém nejen z hlediska porozumění samotného tématu. V oblasti algoritmizace a programování je vhodné znát alespoň základy angličtiny, což může uživateli usnadnit práci.

Posledním kritériem je **cena**. V rámci tohoto kritéria je zásadní, zda je aplikace zdarma, zda obsahuje nějaké prémiové funkce a jaká jsou omezení v závislosti na ceně.

4.1 SoloLearn

První aplikací, na kterou se zaměříme, je SoloLearn. Co se týče dostupnosti, uživatel může ke kurzům přistupovat odkudkoliv. Přístup je možný prostřednictvím webového prohlížeče nebo mobilní aplikace, která je k dispozici ke stažení jak v App Store, tak Google Play. Pro používání aplikace je potřeba se registrovat. K dispozici jsou tři možnosti. Uživatel si buď vytvoří nový účet, nebo může použít svůj účet Facebooku nebo Googlu. Možnost použít existující účet je výhodou, neboť každý uživatel bude mít účet alespoň v jedné ze zmíněných sociálních sítí. Uživatel tak není odrazen nutností vytvořit si nový speciální účet. Nabídka kurzů je skutečně široká. Na výběr je z těchto programovacích jazyků: Python, C, C++, C#, JavaScript, Java, PHP, SQL, HTML, CSS, Ruby a Swift. Některé kurzy jsou zaměřeny speciálně na práci s knihovnami těchto jazyků, a proto by měl uživatel nejprve absolvovat základní kurz.

V aplikaci uživatel prochází různými tematickými celky zaměřenými např. na cykly podmínky, funkce, OOP atd. Může sledovat svůj postup v kurzu (viz Obr. 3). Přístup k dalším lekcím je podmíněný splněním všech úkolů v předchozí lekci.

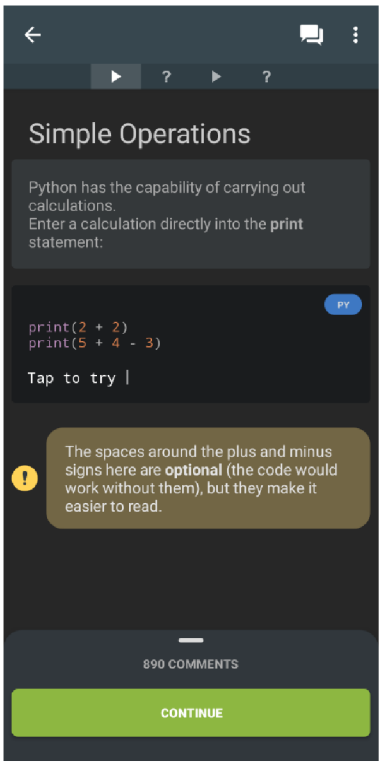


Obr. 3: Přehled úkolů kurzu Python Core z mobilní aplikace SoloLearn

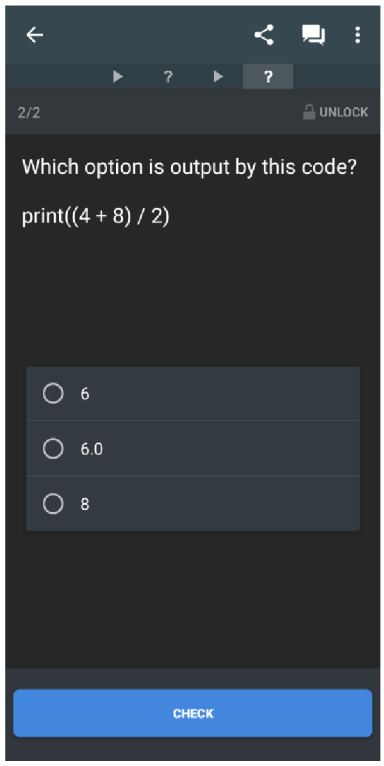
(Zdroj: vlastní snímek obrazovky mobilního zařízení)

Na konci kurzu je možné si nechat vygenerovat certifikát, který lze stáhnout do mobilního zařízení nebo počítače. Uživatel získá odměnu, která ho může motivovat k dalšímu studiu a splnění dalších kurzů. Motivující může být také získávání tzv. odznaků (Badges) za dílčí úspěchy. Za každou dokončenou aktivitu navíc uživatel získává zkušenostní body (XP), které se neustále sčítají a posouvají jej v žebříčku komunity.

Úroveň interaktivity lze označit za vysokou. Jednotlivé lekce mají srozumitelné učební texty a nabízí možnost si rovnou vyzkoušet funkčnost příkladu kliknutím na „Tap to try“ (viz Obr. 4). Najdeme zde mnoho interaktivních úkolů, jako je například přidání chybějícího kódu, seřazení kódu do správného pořadí nebo označení správné odpovědi (viz Obr. 5). Velkou výhodou je to, že je aplikace propojena s komunitou dalších uživatelů. Pro každý kurz je vytvořeno interní fórum, kde se diskutuje o řešeních pro zadané úkoly atd. Uživatel může také využít možnosti tzv. hřiště (code playground) a vyzkoušet si svůj vlastní kód, který pak může být uložen v profilu uživatele.



Obr. 4: Interaktivní úkol kurzu Python Core z mobilní aplikace SoloLearn
(Zdroj: vlastní snímek obrazovky mobilního zařízení)



Obr. 5: Interaktivní úkol kurzu Python Core z mobilní aplikace SoloLearn
(Zdroj: vlastní snímek obrazovky mobilního zařízení)

Webová varianta platformy SoloLearn je pouze v angličtině. Mobilní aplikaci je možné přepnout na jiný jazyk. Konkrétně jsou na výběr tři jazyky: angličtina, ruština a španělština.

Webová i mobilní varianta jsou k dispozici zdarma. Mobilní aplikace navíc umožňuje zakoupení verze PRO. Rozdíly oproti bezplatné verzi jsou následující: žádné reklamy, možnost nastavení denních cílů, možnost najít uživatele ze stejné lokality, možnost vidět, kdo si prohlížel můj profil atd.

SoloLearn je velmi dobře navržená aplikace, jak ve webové, tak mobilní verzi. Návrh aplikace svědčí o tom, že je vytvářena primárně pro mobilní zařízení. Je v ní k nalezení spousta zajímavých kurzů (zejména pro začátečníky). Je k dispozici pro všechna zařízení, takže ji lze využívat odkudkoliv. Vybrané kurzy lze dokonce stáhnout do svého digitálního zařízení, aby byly přístupné off-line. Další výhodou je, že SoloLearn má podobu sociální sítě. Pro některé uživatele tak může být učení zajímavější a zábavnější (sdílení úspěchů s ostatními, diskuze atd.). Pokud budou ve spojení s ostatními, mohou díky tomu mít pocit, že se neučí sami.

4.2 Codecademy

Další aplikací, na kterou se zaměříme, je Codecademy. Aplikace je k dispozici jak prostřednictvím webového prohlížeče, tak v mobilních obchodech s aplikacemi. Mobilní aplikace se jmenuje Codecademy GO. Registrace a následné přihlášení nebo přihlášení pomocí existujícího účtu je vyžadováno pro mobilní i webovou aplikaci. Uživatel si může vytvořit nový účet v rámci platformy Codecademy nebo použít již existující účet Facebooku, Googlu, LinkedInu nebo GitHubu. Mobilní aplikace umožňuje k přihlášení použít pouze první dva zmíněné účty.

Jako v předchozí aplikaci je rozsah nabízených kurzů velmi široký. Nabízí tyto programovací jazyky: HTML a CSS, Python, JavaScript, Java, SQL, Bash/Shell, Ruby, C++, R, C#, PHP, Go, Swift a Kotlin. Nebo si lze vybrat z těchto tematicky zaměřených kurzů: Tvorba webových stánek, Databáze, Počítačová věda, Vývojové nástroje, Strojové učení, Základní principy kódu, Webový design, Vývoj her, Vývoj mobilních aplikací, Vizualizace dat.

Uživatel si může pro představu vybrat konkrétní programovací jazyk, který se chce naučit (procvičit), nebo si může vybrat tematické zaměření.

Když si uživatel vybere požadovaný programovací jazyk (např. Python), zobrazí se mu na výběr několik dalších možností, tzv. dovednostních cest (Skill Paths), např. „Analýza dat pomocí Pythonu“, „Základy Pythonu“ atd.

Pokud si uživatel vybere nějaké tematické zaměření (např. „Tvorba webových stránek“), bude mít také na výběr z dalších možností, tentokrát tzv. kariérních cest (Carrer Paths). Konkrétně kariérní cesta „Vývoj webu – ovládněte jazyky HTML, CSS, JavaScript a SQL“ pak obsahuje další dovednostní cesty, jako je „Tvorba webových stránek pomocí HTML“. Uživatel si tedy může přímo vybrat cestu, kterou se chce vydat (co ho láká) a ani nemusí znát všechny specifické programátorské znalosti spojené s touto cestou. Codecademy to dělá za uživatele a nastaví kurz podle potřeby.

Poslední kategorií s výběrem možností jsou „Kurzy“ (např. „Naučte se JavaScript – základy pro front-end i back-end“). Bohužel kategorie „Kariérní cesta“ a „Dovednostní cesta“ jsou k dispozici pouze uživatelům, kteří si zaplatí Codecademy PRO. Je zde omezený počet lekcí, které jsou k dispozici zdarma, pro většinu je ale vyžadována verze PRO. Po kliknutí na tlačítko „Vyzkoušet PRO zdarma“, si uživatel může vyzkoušet Trial verzi po dobu jednoho týdne. Po uplynutí zkušební doby je vyžadován poplatek. PRO verze je dostupná pouze prostřednictvím webového prohlížeče, ne jako mobilní aplikace.

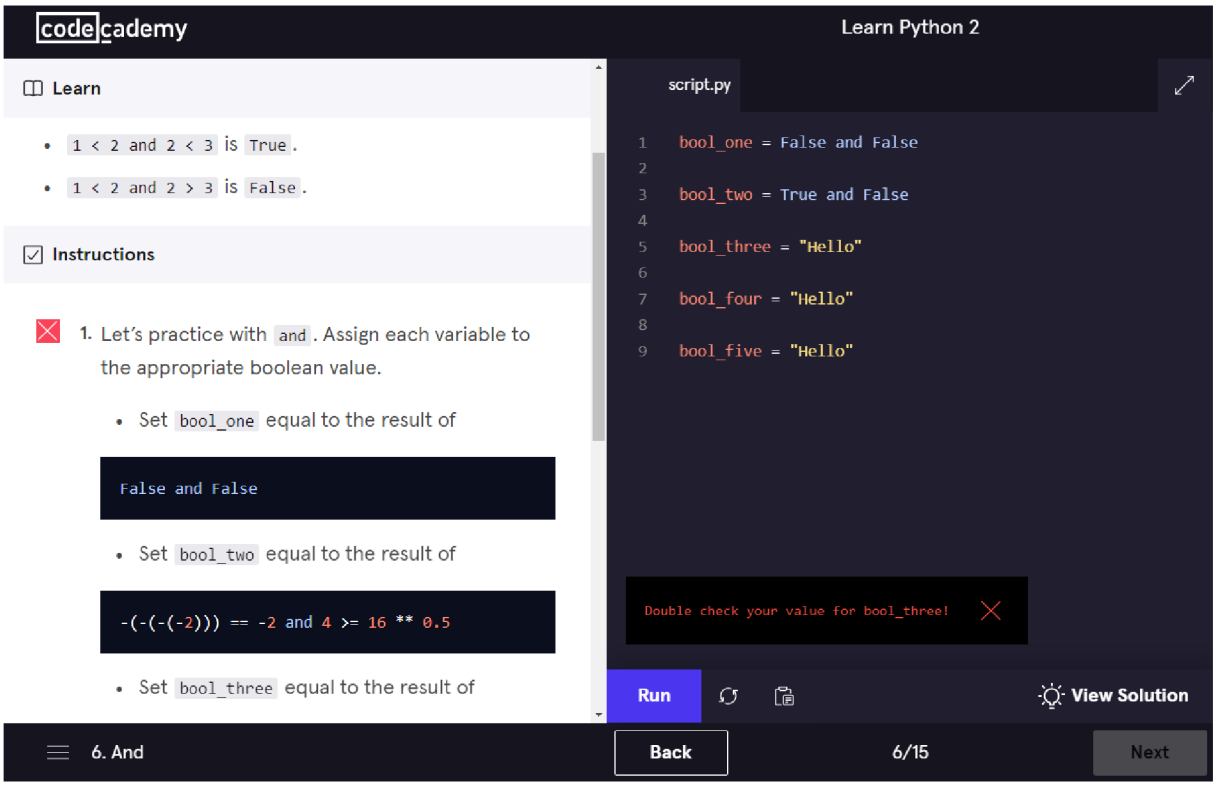
Každý uživatel si může zobrazit učební plán kurzu. Zjištění, že většina lekcí je k dispozici pouze ve verzi PRO, může velmi rychle odradit od zahájení studia přes tuto platformu. Je pravda, že začít zdarma a naučit se nějaké základy může opravdu každý. Bohužel nakonec zjistí, že aplikace je ve své základní verzi natolik omezená, že pokud si ji neplánuje zaplatit, nebude tou správnou volbou.

Úroveň interaktivity lze opět popsat jako vysokou. Mobilní aplikace je ideální pro procvičování. Kurzy a lekce jsou přizpůsobeny mobilnímu zařízení a úkoly vypadají hezky (viz Obr. 6) – např. doplňování neúplného kódu, výběr z nabídky více možností apod.

Obr. 6: Interaktivní úkol z mobilní aplikace Codecademy

(Zdroj: vlastní snímek obrazovky mobilního zařízení)

Lepších výsledků lze dosáhnout při učení prostřednictvím webové aplikace. Velmi přínosnou funkcí je konzole pro vývoj (viz. Obr. 7), ve které uživatel píše kód podle zadání v levé části obrazovky. K dispozici je také učební text k danému tématu. Pokud uživatel napíše kód správně, dokončí úkol a může pokračovat k dalšímu. Tento způsob výuky se snaží simulovat skutečnou práci programátora.



Obr. 7: Interaktivní úkol z webové aplikace Codecademy

(Zdroj: vlastní snímek obrazovky)

Codecademy také obsahuje diskusní fórum, kde lze diskutovat s ostatními uživateli. K motivaci uživatelů se požívají tzv. úspěchy (Achievements), odznaky (Badges) a body (Points).

Jazyková lokalizace v mobilní i webové aplikaci je pouze angličtina.

Codecademy je dobře navržená aplikace, která má uživateli v oblasti algoritmizace a programování co nabídnout. Největší slabinou této platformy je malá nabídka bezplatných kurzů a funkcí. Pokud se uživatel chce ve svých znalostech posouvat dál, je nutné si zaplatit verzi PRO.

4.3 The Odin Project

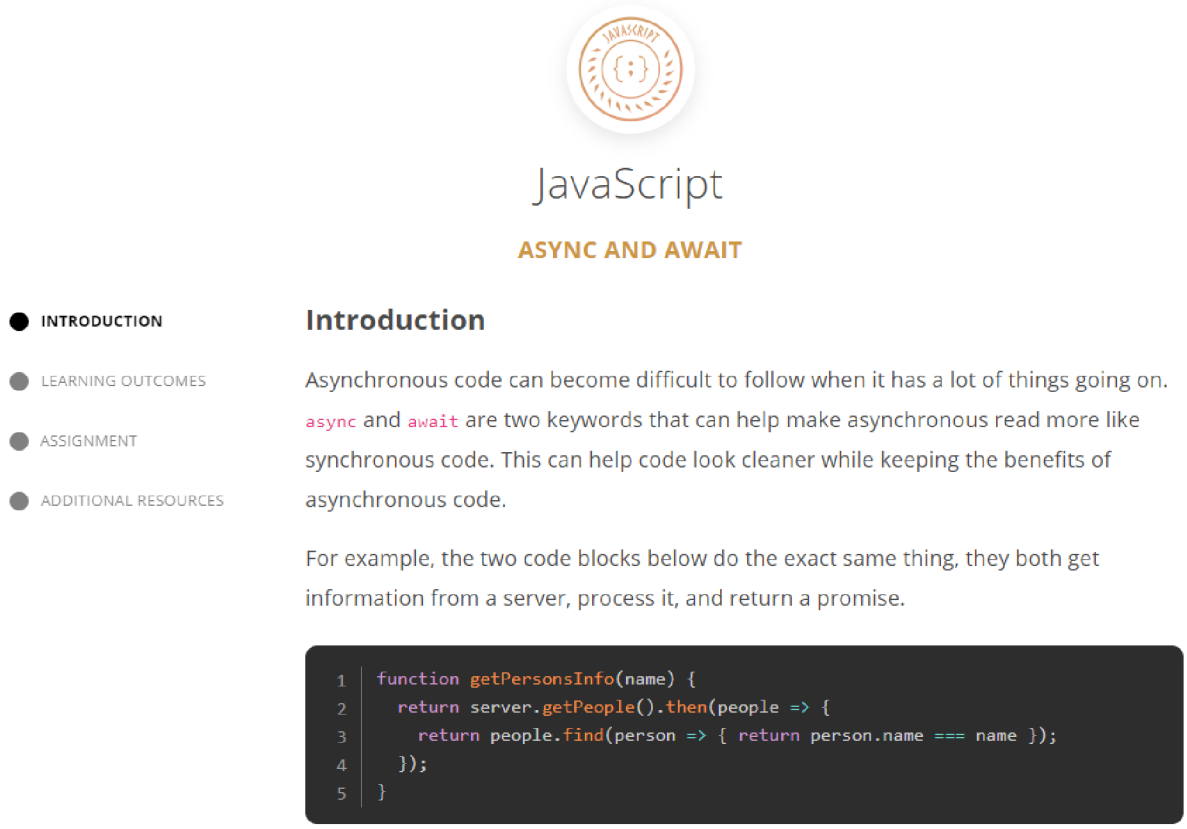
Třetí analyzovanou aplikací je The Odin Project, která existuje pouze ve formě webové aplikace. Dostupná je tedy pouze prostřednictvím webového prohlížeče. Nicméně má responzivní webový design a je plně použitelná i na menších obrazovkách (např. mobilní telefon).

Filozofií tohoto projektu je, že učení by mělo být přístupné každému a zdarma. Cílem aplikace je naučit uživatele vytvářet vlastní webové aplikace. Toho lze dosáhnout

absolvováním všech tematických kurzů. Ty se věnují těmto programovacím jazykům: HTML, CSS, JavaScript, Ruby, SQL.

Do aplikace se lze opět přihlásit svým účtem, ale není to podmínka. Uživatel si může vytvořit nový účet nebo použít již existující účet Googlu nebo GitHubu.

Učení nemá příliš interaktivní podobu ve srovnání s předchozími aplikacemi. Nenajdeme zde žádné úkoly typu „Doplň chybějící část kódu“ nebo „Napiš správný kód do konzole“. Výuka je spíše textového charakteru, doplněná ukázkami kódu a hypertextovými odkaz na sekundární zdroje pro další studium tématu (viz Obr. 8).



Obr. 8: Studijní text z webové aplikace The Odin Project

(Zdroj: vlastní snímek obrazovky)

Výhodou je možnost propojení s komunitou ostatních uživatelů. Na výběr je mezi diskusním fórem v rámci The Odin Project nebo chatovací skupinou v komunikačním systému Discord. Autoři projektu vnímají možnost komunikace a spolupráce s ostatními uživateli jako zásadní.

V tomto případě je obtížnější určit míru interaktivity, vzhledem k tomu, že uživatel musí provádět úkoly v externím softwaru a výukový systém funguje pouze na textové bázi

s vloženými hypertextovými odkazy a ukázkovými částmi kódu. Nicméně aplikace nabízí diskusní fórum a chat. Proto by se interaktivita dala označit jako střední.

Veškeré studijní materiály jsou k dispozici prostřednictvím webové služby GitHub a podléhají licenci Open Source. To znamená, že kdokoliv může přidávat nové funkce nebo opravovat nalezené chyby. Všechny součásti kurzů jsou opravdu zdarma a uživatel není vyzýván k zakoupení jakéhokoliv produktu, jako tomu bylo v předchozích případech.

Webová aplikace The Odin Project je kompletně v angličtině, včetně jejich součástí.

The Odin Project je velmi zajímavá aplikace, kterou lze doporučit všem, kteří chtějí získat nové znalosti z oblasti vývoje webových aplikací. K největším nevýhodám patří absence interaktivních úloh a nutnost provádět úlohy externě ve specializovaném softwaru. Ve skutečnosti jde o žádaný postup z hlediska programátorské praxe, který uživatele více připravuje na reálnou práci. Z hlediska atraktivity a dnešních nároků na rychlost učení to však může některé uživatele odradit.

4.4 freeCodeCamp

Další aplikací je freeCodeCamp. Slovo „free“ v názvu naznačuje, že učení pomocí této aplikace bude zdarma, a je tomu skutečně tak. Není třeba platit za žádný prémiový obsah, uživatel má přístup ke všem kurzům zdarma.

Aplikace je dostupná pouze prostřednictvím webového prohlížeče. Mobilní aplikace není k dispozici. Pro studium to však není překážkou. Stránky jsou plně responzivní a dobře odladěné pro mobilní zařízení. Stačí přístup k internetu a uživatel může studovat odkudkoliv a z jakéhokoliv zařízení.

Do aplikace se můžeme přihlásit čtyřmi způsoby: pomocí účtů Googlu, GitHubu, Facebooku nebo svého e-mailu. Ale opět to není nutné. Přihlášení umožní uložení pokroku v kurzech. Profil uživatele může být veřejně viditelný, aby bylo možné sdílet portfolio dovedností a certifikátů získaných v rámci aplikace.

Učební plán je opět velmi rozsáhlý a je rozdělen do několika sekcí a podsekcí. Mezi hlavní sekce patří: Responzivní webdesign, JavaScript a datové struktury, Front-endové knihovny, Vizualizace dat, Rozhraní a mikroslužby, Datová analýza, Bezpečnost dat, Strojové učení v Pythonu. Každá sekce vždy obsahuje 5 a více podsekcí. Kromě těchto oblastí si lze rozšířit své znalosti prostřednictvím dalších bezplatných kurzů. Vyhledávací políčko nám nabídne více než 7000 dalších kurzů. A počet kurzů se stále zvyšuje. Možnosti učení jsou tedy skutečně velmi rozsáhlé.

Úroveň interaktivity je vysoká. Na obrázku č. 9 je vidět rozložení prvků ve webové aplikaci, které je podobné jako u Codecademy.

Obr. 9: Interaktivní úkol z webové aplikace freeCodeCamp

(Zdroj: vlastní snímek obrazovky)

Každá lekce se skládá z několika úkolů. Když je úkol splněn, zobrazí se oznámení s hodnotou procentuálního progresu a uživatel pokračuje na další úkol. Pracovní prostředí každého úkolu se skládá z několika částí. První částí jsou učební texty, kde se uživatel dozví potřebné informace o tématu. Uživatel zde také nalezne zadání. Další částí je integrovaná vývojová konzole, do které uživatel zadává své řešení úkolu. Pod konzolí se zobrazují informace o správnosti či nesprávnosti uživatelova řešení. Tlačítkem „Run the Tests“ se spustí kontrola kódu a uživatel je informován, zda se v něm vyskytuje nějaká chyba. Kromě možnosti spuštění kódu je k dispozici také reset nebo nápověda ve formě videonávodu nebo diskuzního fóra. Poslední částí je náhledové okno, kde si uživatel může prohlédnout skutečný výstup kódu.

Tento způsob učení je nejvhodnější, protože si uživatel osvojí programovací techniky přímo prostřednictvím praktických ukázek. Musí si dávat pozor na chyby při psaní, správnou syntaxi a sémantiku. Přechod ke skutečnému programování do různých vývojových nástrojů tak bude plynulý a zároveň optimální z hlediska pokroku v oblasti dalšího vzdělávání. Pro uživatele je k dispozici také diskuzní fórum, které může být pro některé klíčovým prvkem při samostudiu.

Jazyková lokalizace je opět pouze anglická.

Aplikace freeCodeCamp nabízí mnoho příležitostí k učení, nejen v rámci předpřipravených oblastí, ale také prostřednictvím dalších tisíců kurzů. Vše je k dispozici zdarma. Lekce jsou dobře navržené a plně využitelné na všech digitálních zařízeních.

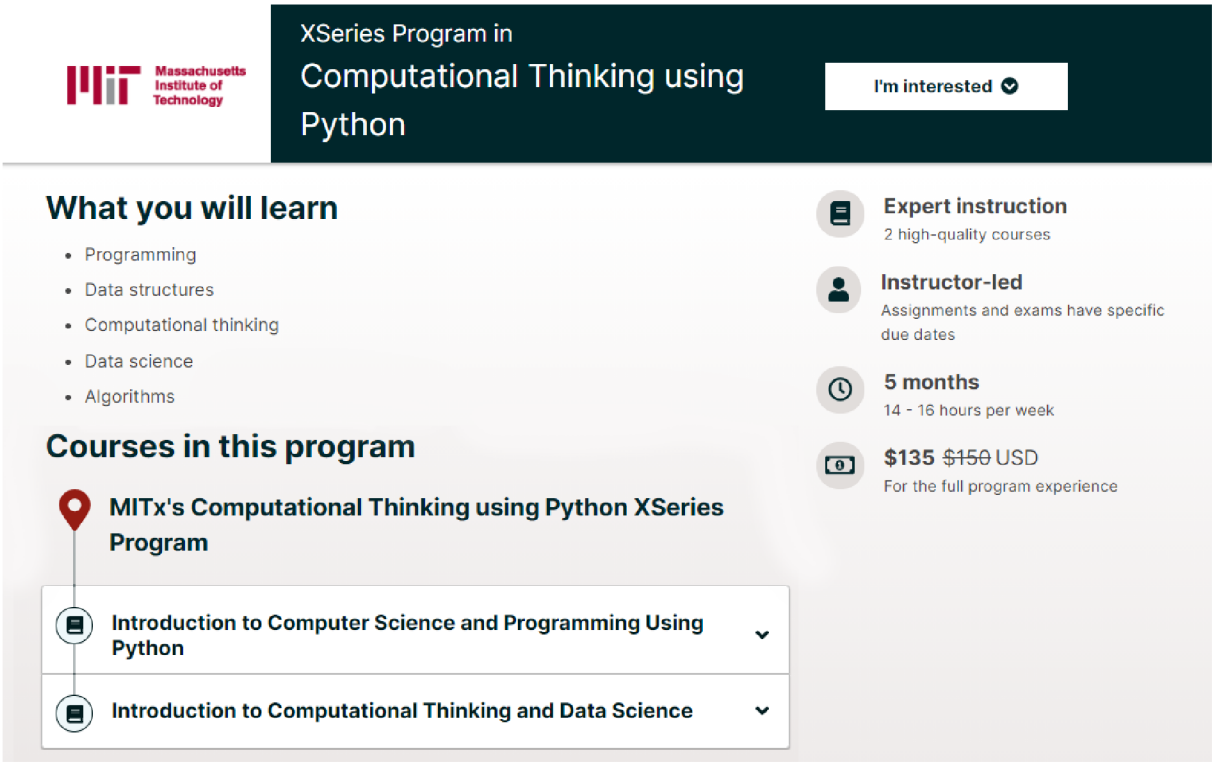
4.5 edX

Platforma edX je přístupná nejen prostřednictvím webového prohlížeče, ale také v App Store a Google Play, kde jsou k dispozici mobilní aplikace. Přihlásit se lze čtyřmi způsoby: vytvořením nového účtu pomocí e-mailu, nebo pomocí již existujících účtů Facebooku, Googlu nebo Microsoftu.

Obsah aplikace edX se netýká pouze programování, jako tomu bylo u předchozích aplikací. Pokrytí témat je opravdu široké. K dispozici jsou nejen kurzy, ale i celé studijní programy z renomovaných světových univerzit a firem. Aby bylo možné porovnat tuto aplikaci s ostatními, zaměříme se pouze na kurzy informatiky. V době psaní tohoto textu jich bylo celkem 810, což je znatelně více než u ostatních oborů. Kurzy zahrnovaly programovací jazyky, jako např. JavaScript, Python, C, C++, R, SQL atd. Některé kurzy lze studovat také v různých jazycích. Převážná většina jich je však v angličtině. Aplikace jako taková nabízí uživateli možnost přepínat jazyk mezi angličtinou a španělštinou.

Kurzy jsou většinou vedeny profesionálními instruktory a po jejich dokončení získá uživatel certifikát, včetně originálního podpisu lektora kurzu. Už podle tohoto popisu lze očekávat, že kurzy nebudou bezplatné a je tomu opravdu tak. Aplikace konstantně uživatele vybízí, aby zaplatil za vybraný kurz (viz Obr. 10). Navíc se kurzy konají v určité termíny a uživatel si nemůže svobodně vybrat, kdy se ho bude zúčastnit. Výuka sestává především z textových a video materiálů. Proto je také úroveň interaktivity nízká.

Mobilní aplikace je navržena na stejném základě jako webová aplikace. Jedinou výhodou této aplikace je, že má uživatel možnost přistupovat k výukovým materiálům off-line. Kvízy jsou v mobilní aplikaci blokovány a obsahují hypertextové odkazy, které uživatele přesměrují do webového prohlížeče. Po přesměrování však zjistí, že ke kvízům nemá přístup, protože nezaplatil poplatek. Tento element je velmi nepříjemný a demotivující.



Obr. 10: Nabídka kurzu ve webové aplikaci edX

(Zdroj: vlastní snímek obrazovky)

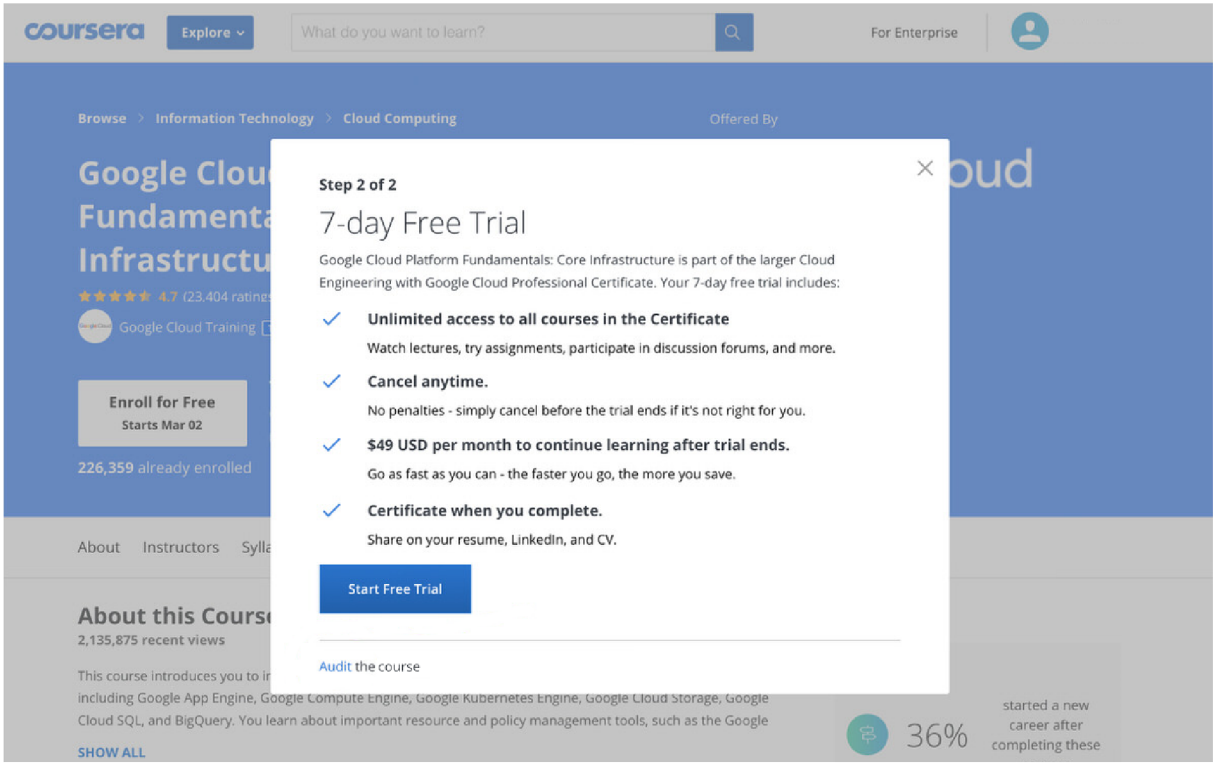
Vzhledem k značným omezením základní verze, tato platforma nebude vhodná pro uživatele, kteří hledají vhodný zdroj vzdělání v oblasti algoritmizace a programování. Aplikace edX může uživatele zaujmout širokou nabídkou kurzů od renomovaných společností a univerzit a některé kurzy jistě mohou být obsahově velmi zajímavé a přínosné. Nicméně různá omezení, která jsou podmíněna zaplacením poplatku (ten se liší v závislosti na kurzu), jsou překážkou a aplikace tudíž není pro běžné uživatele zcela ideální.

4.6 Coursera

Coursera se stejně jako edX zaměřuje na širokou škálu oblastí vzdělávání. K dispozici jsou různé studijní programy a kurzy od známých univerzit a firem. Z oblasti informatiky bylo na platformě v době psaní textu dostupných 1544 kurzů. Mezi vyučované programovací jazyky patří Java, C++, JavaScript, Python aj. Aplikace je kompletně v angličtině, ale některé kurzy jsou i v jiných jazykových lokalizacích.

Nyní se dostáváme k hodnotícímu kritériu, kterým je cena. A stejně jako u edX, je tohle hlavní nevýhodou aplikace. Princip je stejný a bez zaplacení poplatku jsou velké části kurzů a další funkce nepřístupné (viz Obr. 11). Jedním z uzamčených prvků jsou kvízy, které slouží k opakování látky. Některé kvízy lze otevřít a odpovědět na otázky, ale již není možné je

zkontrolovat. To znamená, že uživatel nedostane zpětnou vazbu, což je velmi nepraktické a demotivující v procesu samostudia. Samotné učení má podobu textových a video materiálů. Uživatel má k dispozici také interní diskusní fórum, které je u každého kurzu.



Obr. 11: Nabídka kurzu ve webové aplikaci Coursera

(Zdroj: vlastní snímek obrazovky)

Uživatel může použít jak webovou tak mobilní aplikaci (dostupné z App Store a Google Play). Může se přihlásit pomocí účtů Googlu, Facebooku, nebo svého e-mailu. Výhodou je, že mobilní aplikace umožňuje zpřístupnit výukové materiály off-line. Nevýhodou, že některé úkoly nelze řešit přímo na mobilním zařízení. Aplikace uživatele vyzve k použití větší obrazovky.

V porovnání s aplikací edX jsou zde drobné rozdíly. Struktura kurzů je podobná. Coursera má lépe navrženou mobilní aplikaci, což může být pro některé uživatele rozhodující. Platforma edX ale nabízí lepší platební podmínky. Rozhodnutí tedy závisí na každém uživateli.

Coursera je vysoce kvalitní zdroj vzdělávacích kurzů. Nevýhody, jako jsou platební podmínky a velké množství uzamčených funkcí v základní verzi však převažují nad výhodami, jako je například dobře navržená mobilní aplikace.

4.7 W3Schools

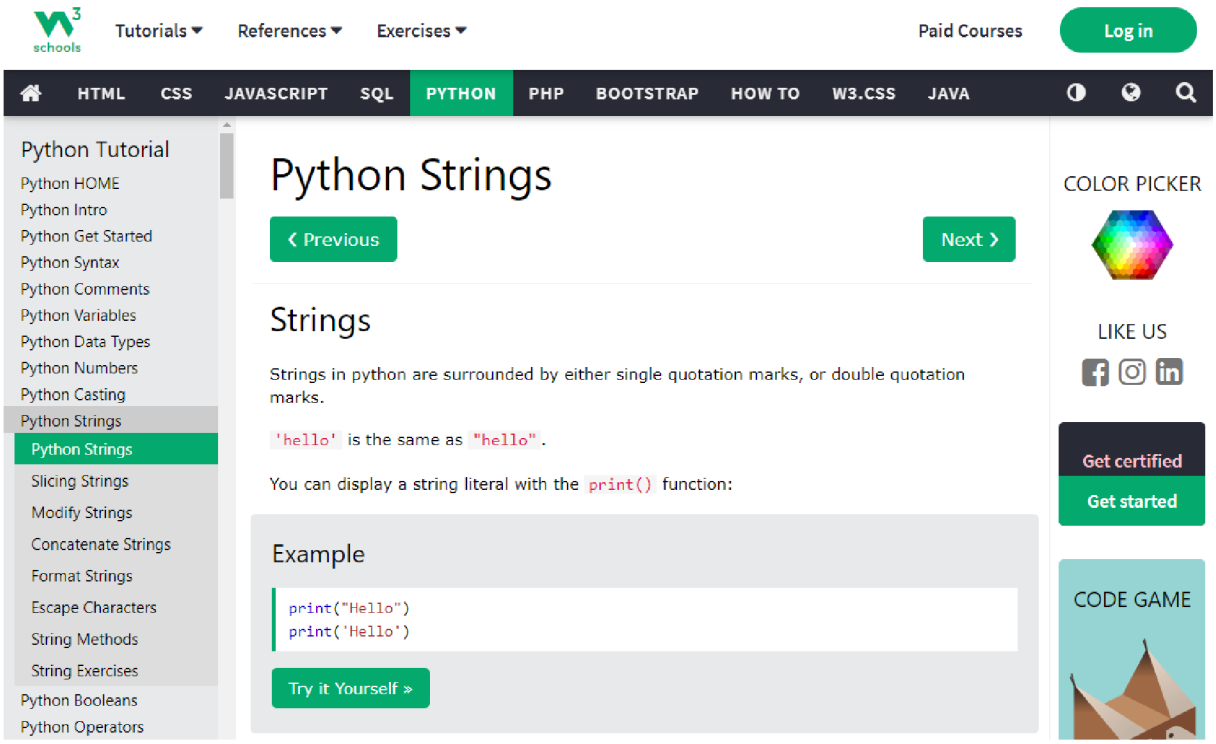
Jedná se o čistě webovou aplikaci, bez speciální mobilní aplikace. Nicméně responzivní design je dobře navržený a nic nebrání tomu, aby mohla být použita i na mobilním zařízení.

Jazyková lokalizace je řešena prostřednictvím implementovaného překladače od společnosti Google, takže lze prostředí přeložit do desítek jazyků pomocí strojového překladu. Nicméně primárním jazykem je angličtina. Obsah kurzů zahrnuje moderní webové technologie a populární programovací jazyky. Mezi ty hlavní patří HTML, CSS, JavaScript, AJAX, Python, Java, C++, C#, SQL, PHP.

Aplikace se skládá ze čtyř částí. První jsou tutoriály (Tutorials), kde si uživatel vybere téma, které chce studovat. Další část reference (References) umožňuje vyhledávat podle tématu a klíčových slov a zobrazit si učební texty daného tématu. Třetí část příklady (Examples) je plná praktických příkladů, využitelných nejen v programování. Poslední část cvičení (Exercises) nabízí uživateli různé možnosti, jak si ověřit své znalosti. V rámci aplikace lze získat certifikáty, které prokazují uživatelovy znalosti ve vybraných oblastech – za poplatek a po absolvování příslušných testů. Všechny ostatní výukové materiály a interaktivní úkoly jsou zdarma.

Úroveň interaktivity je vysoká. V průběhu celého procesu učení obsahuje text menší interaktivní úkoly, příklady kódu atd. Velkou výhodou je možnost využití integrované konzole, která se otevře po kliknutí na tlačítko „Try it Yourself“ (viz Obr. 12). V konzoli si uživatel může příslušný kód vyzkoušet v praxi a libovolně upravovat. Uživatelé mají k dispozici také diskusní fórum.

Výuka informatiky prostřednictvím této aplikace je velmi rozmanitá. Aplikace může sloužit nejen studentům k rozšíření jejich znalostí ve vybrané oblasti, případně k procvičení učiva, ale také učitelům jako zdroj inspirace pro další výuku.



Obr. 12: Studijní text z webové aplikace W3Schools

(Zdroj: vlastní snímek obrazovky)

4.8 Mimo

Aplikace Mimo je k dispozici ve formě mobilní aplikace pro zařízení se systémem iOS (App Store) a zařízení se systémem Android (Google Play) a také prostřednictvím webového rozhraní pro všechna digitální zařízení s přístupem k internetu. Uživatel se může přihlásit pomocí e-mailu, nebo existujících účtů Applu, Googlu nebo Facebooku.

Největším problémem aplikace je opět ustavičné nabízení upgradu na verzi PRO, tzn. placení předplatného za odemknutí prémiového obsahu a funkcí. V základní verzi FREE se uživatel dostane pouze k první položce dané kategorie. Ty, které jsou označeny štítkem PRO, povolí pouze zobrazení názvu lekce. Uživatel základní verze nemůže navíc využívat tzv. hřiště (Playground).

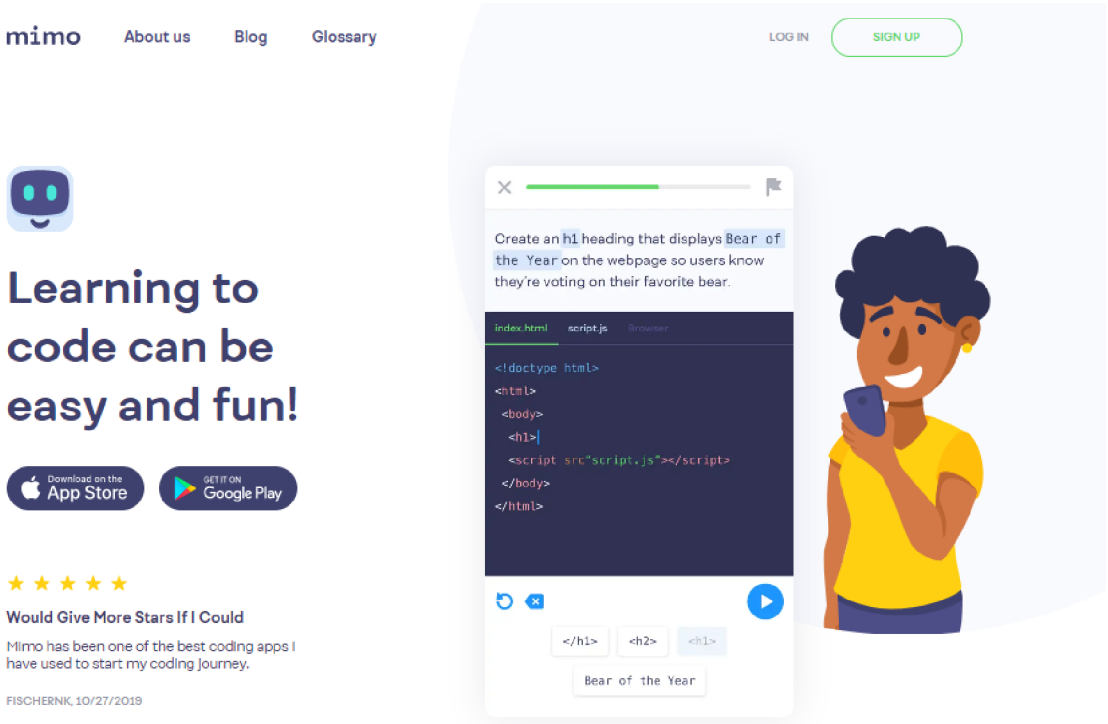
Obsah nabízených kurzů je různorodý. Uživatel má na výběr ze dvou hlavních cest – „Vývoj webu“ a „Programování v Pythonu“, nebo si může samostatně procházet další kurzy, které aplikace nabízí. Jedná se tyto další programovací jazyky: Ruby, Java, Kotlin, C++, C#, Swift, React, PHP, JavaScript, SQL a R.

Úroveň interaktivity je vysoká. Aplikace je graficky pěkně navržená. Kurzy jsou doplněny o interaktivní úkoly. Je zde možnost propojení s programátorskou komunitou

a soutěžení v jejím rámci o pozici v týdenním žebříčku. Další motivační funkcí je např. udělování bodů za splnění úkolů.

Všechny verze aplikace jsou kompletně v angličtině.

Aplikace Mimo je atraktivní jak vzhledem (viz Obr. 13), tak nabízeným obsahem. Design splňuje nejnovější trendy a požadavky komunity. Nevýhodou a velkou překážkou je obecně absence kurzů, které jsou k dispozici kompletně zdarma a z toho vyplývající nutnost přechodu na verzi PRO. Cenová politika je poměrně vysoká a značně omezuje šance aplikace na využití v hodinách informatiky, natož v rámci samostudia.



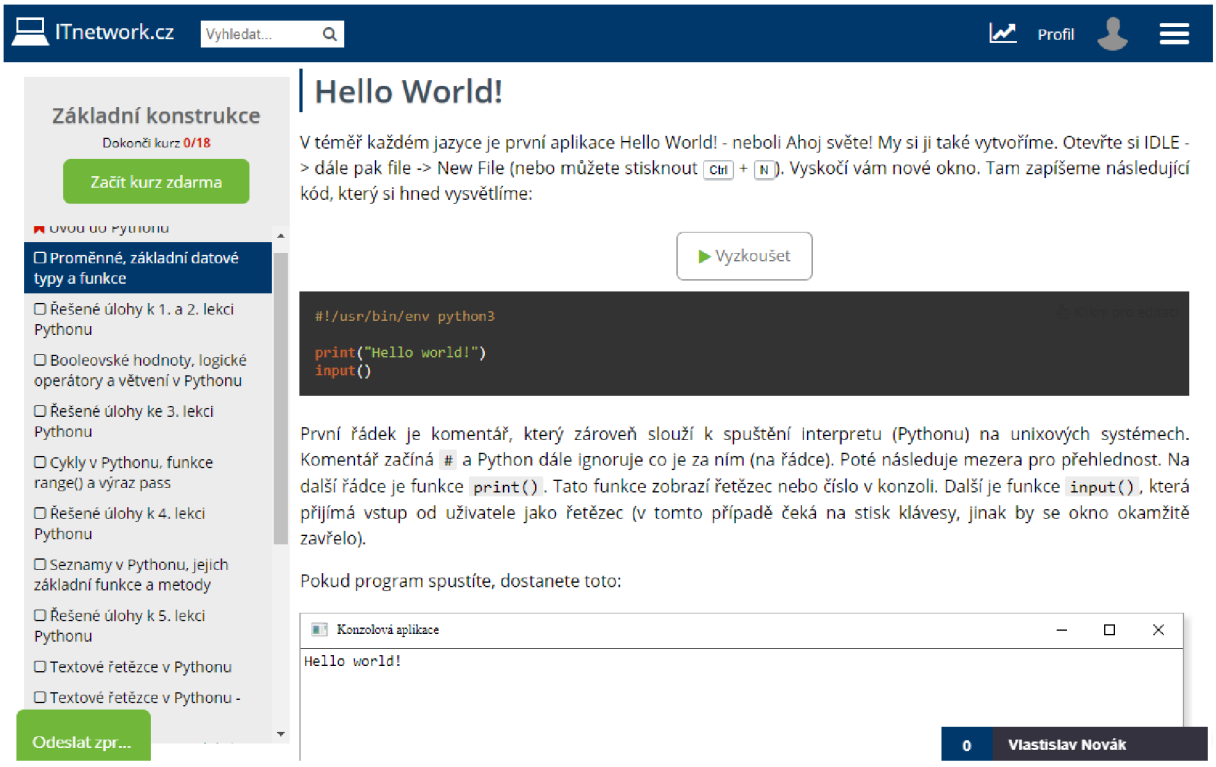
Obr. 13: Úvodní obrazovka webové aplikace Mimo

(Zdroj: vlastní snímek obrazovky)

4.9 ITnetwork.cz

Poslední aplikace je ryze česká. Jedná se v podstatě o sociální síť zaměřenou především na programátory. Nabízí velké množství článků a výukových kurzů pro aktuálně nejpopulárnější programovací jazyky, jako jsou C#, JavaScript, Java, Python, PHP, Swift, Kotlin aj. Má čistě webové rozhraní, bez mobilní aplikace, ale díky responzivnímu designu, je dobře použitelná i na mobilních zařízeních. Přihlásit se lze pomocí e-mailu nebo existujícího účtu Facebooku.

Učební texty v kurzech jsou v českém jazyce a jsou protkány názornými ukázkami v podobě spustitelného kódu (viz Obr. 14). Text obohacují snímky obrazovky, např. vývojového prostředí nebo výsledného programu. Na začátku lekce je vždy krátké shrnutí předešlého probíraného učiva a na konci si uživatel může v archivu stáhnout kompletní zdrojový kód. Je znát, že autoři jsou profesionálové ve svém oboru a vše je hezky vysvětleno. V každé lekci je problematika detailně vysvětlena, nezatížená zbytečnostmi nebo příliš složitými konstrukty.



Obr. 14: Studijní text z webové aplikace ITnetwork.cz

(Zdroj: vlastní snímek obrazovky)

Lekce jsou obecně zdarma, takže jsou velmi dobře použitelné ke studiu programování. Nicméně i ITnetwork.cz nabízí nadstandardní verzi PRO, která obsahuje další příklady k procvičení, certifikát po splnění kurzu a možnost zaslání svého řešení úkolu ke kontrole odborníkovi. Aplikace však upgrade uživateli přímo nevnucuje.

Kurzy jsou kvalitně zpracovány, srozumitelně sepsány a doplněny o ilustrační obrázky. Jsou rozděleny do kategorií podle zkušenosti uživatele, takže pomohou jak naprostým začátečníkům, tak pokročilejším programátorům. Součástí všech kurzů jsou navíc diskusní fóra. Neobsahují však interaktivní konzoli, kde by si uživatel mohl testovat svůj vlastní kód, a výukové texty působí staticky. Úroveň interaktivity je tak spíše střední.

4.10 Zhodnocení

V této podkapitole zhodnotíme všech devět vybraných aplikací, které lze využít při rozvoji IM v oblasti algoritmizace a programování.

Všechny aplikace měly svou webovou verzi a některé také speciální mobilní verzi. Obsah jednotlivých kurzů byl velmi podobný. Můžeme zde najít překryv mezi oblíbenými programovacími jazyky, jako jsou Python, JavaScript, PHP atd. Pouze dvě aplikace přesahovaly rámec algoritmizace a programování a nabízely jiné tematické oblasti vzdělávání. Úroveň interaktivity se pohybovala od nízké po vysokou a pokryla tak všechny stanovené úrovně. Většina aplikací byla v anglickém jazyce, jen některé podporovaly jiné jazykové mutace. Cena se ukázala být rozhodujícím kritériem, neboť radikálně ovlivňovala dostupnost studia.

Na základě zjištěných informací se mezi aplikace vhodné k zařazení do výuky dostaly tři ze všech hodnocených. Jako nejatraktivnější se jeví aplikace **SoloLearn**, která umožňuje studium na všech digitálních zařízeních. Mobilní aplikace dokonce umožňuje přístup ke kurzům off-line. Výhodou je také podobnost se sociální sítí, tedy propojení komunity uživatelů pomocí diskuzních fór, porovnávání a další prvky. Nabídka kurzů je dostatečná a uspokojí každého. Tato aplikace si ze všech ostatních vzala to nejlepší a zaslouženě je králem mezi vzdělávacími aplikacemi z této oblasti.

Druhou aplikací je **freeCodeCamp**. Široká nabídka bezplatných kurzů a dobře sestavený učební plán patří k jejím největším přednostem. Jedinou výtkou může být absence mobilní aplikace. V prohlížeči mobilního zařízení však funguje více než obstojně, takže by možná tvorba speciální mobilní aplikace byla kontraproduktivní.

Třetí aplikací je **W3Schools**, která se tváří více jako klasická webová stránka s výukovými programy. Nicméně množství interaktivních prvků z ní dělá plnohodnotnou vzdělávací platformu. Možnosti výuky a následného procvičování vybraných programovacích jazyků jsou nekonečné. Všechny výukové materiály a interaktivní úlohy jsou zdarma. A tato vlastnost je, zvláště mezi programátory zvyklými na sdílnou a spolupracující komunitu, ceněná. Z tohoto důvodu je velmi slibný i projekt The Odin Project, který se stále vyvíjí a záleží jen na komunitě, kam až ho dostanou.

Tři výše zmíněné aplikace mohou být užitečné při výuce informatiky nejen pro žáky, ale i pro samotné učitele. Obecný přehled všech aplikací analyzovaných výše v kontextu stanovených hodnotících kritérií lze nalézt v tabulce č. 3.

Tab. 3: Srovnání analyzovaných aplikací pro rozvoj IM

Název	Dostupnost		Obsah	Interaktivita	Jazyk		Cena
	Web	Mobil			Web	Mobil	
SoloLearn	✓	✓	**	***			Zdarma + PRO
Codecademy	✓	✓	**	***			Zdarma + PRO
The Odin Project	✓	✗	*	**		✗	Zdarma
freeCodeCamp	✓	✗	****	***		✗	Zdarma
edX	✓	✓	***	*			Zdarma + PRO
Coursera	✓	✓	***	***			Zdarma + PRO
W3Schools	✓	✗	**	***		✗	Zdarma
Mimo	✓	✓	**	***			Zdarma + PRO
ITnetwork.cz	✓	✗	**	**		✗	Zdarma

(Zdroj: vlastní zpracování)

5 Výuka algoritmizace a programování

Algoritmizace a programování k sobě v podstatě neodmyslitelně patří, protože programování, tedy psaní konkrétního kódu, následuje vždy až po algoritmickém rozložení problému. K pochopení problematiky je potřeba si nadefinovat základní pojmy, které by měli všichni učitelé této oblasti znát.

5.1 Definice pojmu algoritmizace

Algoritmizace je metodický přístup k vytváření programu. Zabývá se formulací postupů řešení daného problému. Výsledkem algoritmizace je algoritmus, což je posloupnost příkazů popisující řešení daného problému (Dohnal, 2009).

Důležitým znakem algoritmizace je její stálost v čase, což je zásadní rozdíl oproti programovacím jazykům. Programovací jazyky časem zastarávají a musí být aktualizovány nebo úplně nahrazeny novějšími a dokonalejšími. Algoritmizace je však na programovacím jazyce nezávislá, je neměnná a je možné ji využívat v různých oblastech informatiky.

Algoritmizaci lze dělit např. do těchto kroků:

- **formulace problému** – spočívá ve formulaci požadavků, určení výchozích hodnot a požadovaných výsledků. Dále je potřeba specifikovat formu a přesnost výsledků.
- **analýza úlohy** – znamená ověření řešitelnosti úlohy a první nástin řešení. Dále zjišťujeme, zda jsou vstupní hodnoty dostačující a zda má úloha jedno nebo více řešení. Vybíráme nejvhodnější a nejefektivnější řešení.
- **vytvoření algoritmu** – sestavení přesného sledu instrukcí, vedoucích k správnému řešení. Algoritmus ukazuje postup řešení, ale nedává konečné odpovědi.
- **sestavení programu** – sestavení programu (zdrojového kódu) v programovacím jazyce na základě algoritmu. Zdrojový kód se následně pomocí překladače převede na spustitelný program. Dobře provedená analýza a algoritmizace je zásadní pro sestavení správného kódu.
- **odladění programu** - poslední fáze, v které se odstraňují případné chyby v kódu. Syntaktické chyby je schopný odhalit již překladač a lze je snadno vyřešit. Logické chyby vyplývají z nesprávného návrhu algoritmu a jejich oprava bývá náročnější. Po odstranění všech chyb lze program využít k praktickému řešení úlohy.

Algoritmus

Algoritmus je konečná posloupnost přesně definovaných instrukcí, obvykle k řešení specifických problémů nebo k provedení výpočtu. Algoritmu jsou dodávána vstupní data a po jeho provedení generuje data výstupní.

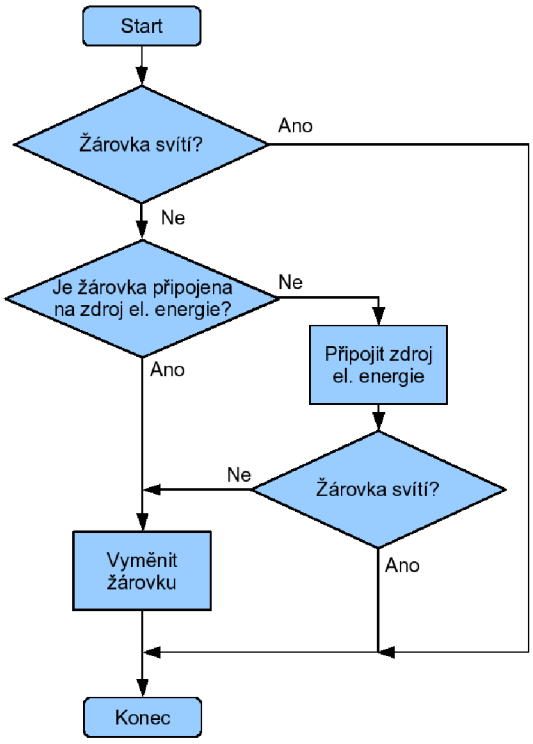
Hlavní výhodou počítače je velmi vysoká rychlost zpracování algoritmů. Všechny postupy výpočtů ale počítači zadává programátor, který je omezen mimo jiné i absencí jakéhokoliv autonomního rozhodování počítače. Je tedy nutné splňovat podmínky definované základními vlastnostmi algoritmů, které jsou

- **determinismus** – je jednoznačně určeno, v jakém stavu se algoritmus nachází v určitém okamžiku a jaký krok bude následovat. Pokud je potřeba se rozhodnout mezi více možnými cestami, musí na základě vstupních dat existovat jen jediná správná možnost. Počítač není schopný se rozhodnout sám za sebe.
- **resultativnost** – vede ke správnému řešení.
- **konečnost** – musí skončit v konečném počtu kroků. To znamená, že se nemůže dostat do nekonečného zacyklení a výsledek musí být dodán v přiměřeném čase. Existují i nekonečné výpočetní metody, které běží neustále a v pravidelných intervalech (např. každý den v určitou hodinu) poskytují výsledky.
- **opakovatelnost** – při zadání stejných vstupních dat musí být výsledek vždy stejný.
- **obecnost** – neřeší pouze jeden konkrétní problém (např. vynásobení 3 a 5), ale obecnou množinu obdobných úloh (např. vynásobení dvou celých čísel); má širší množinu možných vstupních dat.
- **přehlednost** – někdy se uvádí jako nutná vlastnost taky přehlednost, která umožňuje pozdější rozšiřování a vylepšování stávajícího algoritmu.

Vytvořený algoritmus lze vyjádřit různými zápisy. Neexistuje norma, která by byla jednoznačně „správná“. Nejjednodušší je vyjádření prostými popisnými větami v daném jazyce. Jelikož může být algoritmus jakákoliv opakující se činnost, vystačíme si s neformálním verbálním popisem.

Mezi grafická znázornění patří vývojové diagramy (viz Obr. 15). Ty jsou založeny na několika základních tvarech (jako obdélník, lichoběžník, kosočtverec, elipsa apod.) z nichž každý má svůj význam. Jednotlivé kroky jsou propojeny šipkami a postup je zpravidla shora dolů. Výsledný zápis by měl být srozumitelný i ne-programátorovi.

Nejsložitější je z pohledu zápisu i čitelnosti jeho zpracování v konkrétním programovacím jazyce. Jedním z jazyků s nejlépe čitelným kódem je např. Python.



Obr. 15: Příklad vývojového diagramu

(Zdroj: https://cs.wikipedia.org/wiki/Vývojový_diagram)

5.2 Definice pojmu programování

Program lze definovat jako „algoritmus zapsaný v některém programovacím jazyce“ (Ďuráková, 2002, s. 28). Takže programování je přepis algoritmu do daného programovacího jazyka. Každý programovací jazyk je ale jiný a vhodný pro řešení jiného typu úloh, nelze tedy konkrétně určit, který je „nejlepší“. Některé algoritmy lze jednoznačně označit za efektivnější než ostatní (na stejnou úlohu stačí menší počet kroků). Navíc různé programovací jazyky se liší v rychlosti vykonání stejné instrukce. Žáci by tedy měli být vedeni k tvorbě, co možná nejjednodušších algoritmů bez zbytečných kroků, zpomalujících běh programu.

Z historického hlediska došlo na poli programovacích jazyků k velkému vývoji, stejně tak jako v metodice programování. První počítače byly velmi pomalé, na výsledek jednoduchého algoritmu se čekalo i hodiny. Kód proto nemusel být nijak přehledný, hlavně musel být napsán tak, aby byl jeho běh co nejúspornější. Vývoj počítačů ale pokračoval rychle kupředu a objevily se také vyšší programovací jazyky, jako např. assembler (neboli jazyk symbolických adres). Ale čím dál složitější a delší programy začínaly být extrémně

nepřehledné. Proto v šedesátých letech vznikla technika zvaná modulární programování, která zdůrazňovala rozdělení velkých projektů na menší nezávislé moduly. Každý modul se specializoval na jediný aspekt požadované funkcionality. S cílem dosáhnout lepší srozumitelnosti a zrychlení vývoje programů vznikla v sedmdesátých letech metodika zvaná strukturované programování. Ta přinesla tzv. top-down metodu programování, bez nesmyslných příkazů „skok“, které značně přispívají k nepřehlednosti (tzv. špagetovému kódu). V devadesátých letech vzniklo objektově orientované programování, které se snaží co nejvíce přiblížit fungování reálného světa a je hlavním proudem dodnes.

5.3 Diskuze o vhodnosti výuky programování

V předchozích kapitolách byly definovány základní pojmy, jako algoritmus a jeho vlastnosti a programování. Tato problematika se některým může zdát příliš komplikovaná a raději se jí vyhnou obloukem. Hlavním argumentem je, že programování vyžaduje vysokou míru abstrakce, které nejsou děti schopny. Na ověření či vyvrácení tohoto tvrzení dokonce vzniklo několik studií.

Překvapivý závěr má studie Saeeda Dehnadiho (2009), který tvrdil, že jsou pouze dvě skupiny žáků, ti co jsou a nejsou schopni se naučit programovat. Na základě svého výzkumu vytvořil test, kterým lze oddělit tyto dvě skupiny žáků. Přestože sám Dehnadi přiznává, že po dvou letech ověřování spolehlivosti testu se nepodařilo prokázat jeho účinnost, u žáků, kteří byli schopni testem projít, byla vyšší pravděpodobnost úspěšného zvládnutí kurzu programování.

Programátor Jeff Atwood v článku „Please Don't Learn to Code“ píše o bývalém starostovi New Yorku Miku Bloombergovi, který v roce 2012 prohlásil, že se naučí programovat. Atwood kritizuje myšlenku, naučit co nejvíce lidí programovat a označuje ji za absurdní. Pro politika, stejně jako každého jiného je mnohem důležitější např. schopnost orientovat se na internetu, než psát počítačový kód. Zdůrazňuje, že tato snaha staví metodu před problém. „Před tím, než začnete programovat, zjistěte, jaký je váš problém. Máte vůbec nějaký? Dokážete ho popsat tak, aby mu porozuměli i ostatní? Zjistili jste si dostatečné množství informací k jeho vyřešení? Potřebujete k tomu opravdu programování?“ (Atwood, 2012). Tento bod se dotýká spíše samotné algoritmizace, kde je nutné umět nejdřív přesně analyzovat problém. Atwood tvrdí, že cílem výuky by tak spíše měla být snaha, naučit žáky obecné postupy a metody aplikovatelný v každodenním životě.

Bývalý UI návrhář Applu, Bret Victor přichází ve svém článku „Learnable Programming – Designing a programming system for understanding programs“ s ještě větší kritikou. Tvrdí, že celá výuka programování je od základů špatně. Míří tím především na zastaralý způsob výuky programování pomocí konkrétního programovacího jazyka, jakým donedávna býval např. Pascal. Podstatou jeho argumentu je, že žák vidí výsledek správně napsaného kódu, ale nevidí rozfázovaný detailní postup, což znesnadňuje jeho pochopení (Bret, 2012).

Opačný argument má Heggart (2014). Podle něj se v žádném případě nevylučují znalosti toho, co lze s počítačem dělat a jak to počítač procesuje, naopak se podporují. Chápe problematiku opačným pohledem, tedy že pochopení práce počítače napomáhá řešení problémů v běžném životě.

Pecinovský (2005) kritizuje konzervativnost učitelů, kteří se nedostatečně věnují tématu objektově orientovaného programování a upřednostňují stále jen procedurální programování. Celý problém dále prohlubují špatně zpracované učebnice, které ukazují jen syntaxi daného jazyka a samotnou algoritmizace opomíjí. Řeší také otázku nejvhodnějšího věku žáků pro začátek výuky programování. Za ideální považuje pátou až šestou třídu. Začínat s programováním např. až na střední škole je podle něj pozdě.

Doc. PaedDr. Jiří Vaníček, Ph.D, který je vyučujícím didaktiky informatiky a dokonce i přímo didaktiky programování na pedagogické fakultě Jihočeské univerzity v Českých Budějovicích se zabývá algoritmizací ve svém příspěvku „Výuka algoritmizace patří především do informatiky“ z roku 2016. Hodiny informatiky podle něj žáky učí převážně uživatelské dovednosti, které však informatické myšlení nijak nerozvíjejí a mají krátkou životnost. Software se neustále vyvíjí a prakticky každý rok přichází společnosti s aktualizovanou verzí, ve spoustě aspektů odlišnou od té předchozí. Přínosnější by bylo, zaměřit se více na algoritmizaci, kde by se žáci naučili „následovat algoritmus, vytvářet a objevovat jej, porovnat, který z algoritmů je podle různých kritérií lepší, nacházet v algoritmech chyby a schopnost vyjádřit jej v nějakém jazyce tak, aby byl bezesporný.“ (Vaníček 2016). Algoritmizace je podle něj komplexní oblast, kterou lze v dostatečné kvalitě učit jak v hodinách informatiky, tak v jiných předmětech.

Jak zmiňuje Jeff Atwood: „učit všechny žáky programovat“ nedává smysl. Algoritmizaci a programování lze připodobnit třeba k výuce matematiky. Málokdo v praxi využije něco jiného než naprosté základy. Jen malé procento žáků se v dospělosti skutečně bude živit programováním. Výborná znalost vyšších programovacích jazyků, jako např. Java nebo C#, jsou vcelku zbytečné. Nicméně mohou těžit ze znalostí základní algoritmizace.

Účelem výuky programování a algoritmizace by mělo být objasnění principů právě algoritmizace, přičemž programování je jen nástroj k dosažení tohoto cíle. Jde tedy především o schopnost umět vymyslet postup řešení daného problému. Mnoho lidí má problémy s řešením komplexnějších situací, ať už se jedná o projekt ve škole, či neobvyklou pracovní výzvu. Jsou paralyzovaní složitostí dané situace a často se zaseknou hned v začátku. Řešením je právě princip IM zvaný dekompozice. Dovednost rozložit problém na posloupnost na sebe navazujících kroků v tomto hraje důležitou roli.

5.4 Cíle a metodika výuky algoritmizace a programování

Cíle výuky algoritmizace na ZŠ formuloval Tomáš Pitner (2000):

- Primárním cílem je naučit algoritmicky myslet, zformulovat zadání problému.
- Problém analyzovat nejdříve dekompozicí – rozložením na podproblémy.
- Nezbytné je též umět myšlenku dovést k formalizovanému návrh algoritmu, nejlépe v grafické podobě – vývojové diagramy, struktogramy…, eventuálně ve formě přesného slovního popisu za použití předem daných „obratů“, tj. vlastně povolených programových struktur.
- Přepsání formalizovaného návrhu do podoby programu je technická záležitost, nikoli hlavní cíl výuky.
- K samotným algoritmům nedílně patří, ale spíše až „ve druhém pořadí“, datové struktury (objekty).
- Vedlejším cílem (na nižších stupních dokonce hlavním cílem) je celkový rozvoj tvořivosti.

Volba metodiky

Pitner (2000) ve své práci navrhuje dva možné přístupy výuky algoritmizace. První je strukturovaný přístup a druhý objektově orientovaný.

Strukturovaný přístup spočívá v definici posloupných kroků, vedoucích k řešení problému. Tato definice vede k sestavení algoritmu a přepisu do programovacího jazyka. Cílem tohoto přístupu je zvládnutí základních algoritmických struktur, jako jsou větvení, cykly atd.

Podstatou **objektově orientovaného přístupu** je metodický předpoklad, že všechno jsou objekty s nějakými vlastnostmi. Řešení problému spočívá v hledání objektů vyskytujících se v reálním světě, definice jejich vlastností a vzájemných vztahů.

Praxe obvykle ukazuje, že není nezbytné, učit rovnou objektově orientovaným přístupem, ale že pokud začneme učit (kvalitně) strukturovaně, nic se nezmešká. Naopak objektově orientovaný přístup vyžaduje vyšší míru abstraktního myšlení. Je navíc pravdou, že většina dnešních výukových nástrojů pro algoritmizaci je strukturovaného zaměření.

Obecně lze doporučit bez ohledu na objektovost/strukturovanost hned od počátku problémově orientovanou výuku s vysokým podílem samostatné práce (Pitner, 2000).

Žáci se s pomocí známých prostředků snaží vyřešit problém, postupně zjišťují, co neumí, učitel jim to ve správnou chvíli odtajní a vysvětlí, tak aby to mohli použít v praxi. Je vhodné hned na začátku vždy podrobně rozebrat určité řešení a napsat si algoritmus.

Rýdlo (2012) ve své práci zmiňuje 13 obecných metod výuky programování:

- **programming from scratch** – z minulosti nejrozšířenější forma. Znamená tvorbu programu od úplného začátku, bez jakéhokoliv předpřipraveného kódu. Student musí vědět, co všechno bude program obsahovat a umět tyto prvky propojit do finálního řešení. Výhodou je, že žák vnímá veškeré části kódu ve vší jeho komplexnosti. Metoda není vhodná pro počáteční fázi výuky, protože je časově náročná. Odvádí od konkrétního dílčího problému, protože nutí nahlížet na program jako na celek.
- **rozšiřování existujícího kódu** – je opakem předchozí metody a je hojně využívána při výuce OOP. Vysvětluje se lépe na příkladu, který obsahuje větší množství tříd. Nedává smysl, aby začátečník tvořil hned rozsáhlou aplikaci. Vyučující tak nachystá ukázkové programy, do kterých student pouze doplňuje části kódu. Výhodou je rychlá práce zaměřená na konkrétní problém. Navíc se tato metoda více blíží reálným podmínkám v prostředí firmy, kde se často pracuje s již existujícím kódem.
- **používání snippetů** – snippety jsou často používané krátké úseky kódu. Používají se k demonstraci syntaxe určitých struktur. Jejich výhodou je reálnost (praktické využití). Nevýhodou pak může být vytržení z kontextu. Proto není vhodné je používat v počáteční fázi výuky.
- **vodopádový model** – sekvenční vývojový proces, který je založený na stupňovitosti vývoje, kdy se jednotlivé fáze nepřekrývají. Obvykle se skládá ze sedmi fází (specifikace požadavků, návrh, implementace, integrace, testování, instalace, údržba). V praxi se používají modernější agilní modely, které zajišťují lepší řízení velkých projektů, nicméně pro jednoduché úlohy je dostačující.
- **agilní programování** – moderní a cyklický přístup. Na rozdíl od vodopádu se jednotlivé fáze mohou a mají opakovat. Klade se důraz na flexibilní reagování na změny, fungující

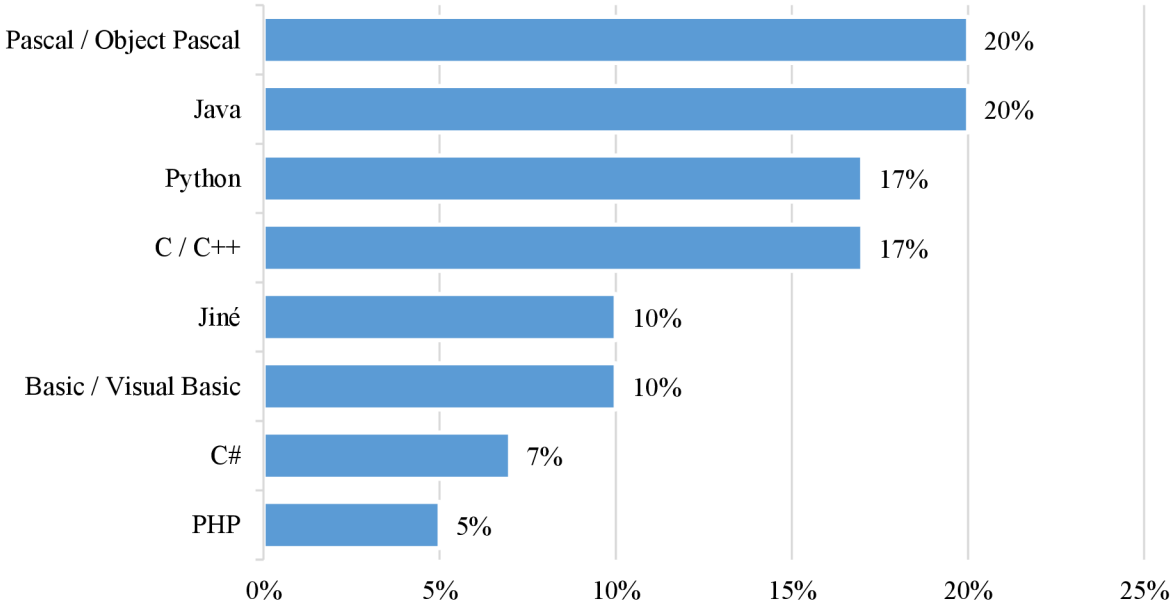
software před vyčerpávající dokumentací, interakci. Tyto hodnoty jsou dobře aplikovatelné ve výuce, kdy žák pravidelně konzultuje svou práci s vyučujícím a pravidelně odevzdává dílčí části, nikoliv až hotové dílo. Vyučující může v průběhu zadání měnit a uzpůsobovat jej didaktickým potřebám. To žáka nutí psát přehledný kód, tak aby se v něm později dokázal vyznat. Zároveň to klade větší nároky na vyučujícího a ochotu žáka věnovat se programování intenzivně.

- **extrémní programování** – metodika vzešlá z agilního programování. Základními principy jsou rychlá zpětná vazba, předpoklad jednoduchosti, přírůstková změna, využití změny a kvalitní práce (Beck, 2002). Je vhodné tyto principy zařazovat do výuky, protože napomáhají pochopení látky. Metodika klade důraz na pečlivé testování, revizi kódu a jeho detailní pochopení, co nejjednodušší implementaci a zaměření na podstatné části.
- **párové programování** – jedna z metod navrhovaných metodologií extrémního programování. Zjednodušeně se dá popsat jako programování dvou programátorů na jednom počítači. Přináší efektivnější vývoj, hlavně díky kolektivní práci. Ten z dvojice, který má zrovna nápad na řešení, píše kód. Druhý se snaží jej pochopit a vnášet vlastní myšlenky. Práce ve dvojici přináší menší míru zodpovědnosti jedince. Pokud žáci nejsou schopni spolupracovat ve dvojici, doporučují se týmy o třech lidech.
- **TDD** – test driven development neboli vývoj řízený testy je také extrémní technikou. Programátor před samotným vývojem vytvoří testy – podmínky, jak má vypadat výstup funkcí pro určité parametry. Pokud funkce projde úspěšně všemi testy, lze ji považovat za hotovou. Ve výuce lze tvorbu testů nechat na učiteli. Student pak vytváří kód a pomocí testů si ověřuje správnost implementace. Testy musí být napsány tak, aby pokryly co největší škálu možností. Existuje totiž riziko, že testem projde kód, který je neefektivní nebo dokonce nesprávný. Přímá kontrola kódu vyučujícím, tak i zde má stále své místo.
- **miniprojekty** – princip spočívá v zadání úkolu skupince žáků. Ti tvoří kompletní program a po dokončení jej prezentují vyučujícímu. Úkol nesmí být příliš komplikovaný, aby jeho řešení netrvalo dlouho, ale zároveň musí otestovat odborné znalosti žáků. Takové úkoly zpravidla pokrývají větší rozsah probraného učiva. Nevýhodou je, že musí mít vyučující zásobu zadání, tak aby se předešlo kopírování předchozích řešení.

- **programování na papír** – metoda, která se často podceňuje. Jde však o výborný způsob jak si procvičit porozumění kódu a zvyšování úrovně faktických znalostí. Učí, méně se spoléhat na překladač do strojového kódu a objevovat syntaktické chyby rychleji. Žák zároveň lépe pochopí přesný význam toho, co píše.
- **chybný kód** – chyby v programech jsou nedílnou součástí vývoje. Proto i žáci by měli být konfrontováni s chybným řešením, ať už vlastním nebo „vzorově špatným“. Většina chyb se opakuje a je dobré se s nimi seznámit. Žáci jsou pak schopni chybu rychleji odhalit a opravit.
- **PBL** – problem based learning je založeno na řešení reálných problémů. Úkoly odpovídající realitě zvyšují porozumění a motivaci. Jedná se o nejvíce využívanou výukovou metodu. Metoda klade velké nároky na samostatnost žáků a studijní materiály.
- **multimediální výuka** – může vhodně nahradit klasický výklad. Multimediální materiály a e-learning přispívají k porozumění lépe, než klasické učebnice. Nicméně stále se doporučuje multimédia zařazovat spíše jako doplněk, než jako jediný prostředek. Ideální se jeví použití online testů a kvízů. Problémem je jejich velká pracnost a časová náročnost, kladená na vyučující.

6 Volba programovacího jazyka

Na základě provedených výzkumů a vlastních zkušeností jsme se rozhodli, že ideálním programovacím jazykem pro rozvoj informatického myšlení bude Python. Podle výzkumu používaných programovacích jazyků na středních školách od Kotka (2013) patří Python k nejpoužívanějším (viz Graf 2).



Graf 2: Převážně používané programovací jazyky (%)

(Zdroj: Kotek, 2013)

Kotek dále uvádí, že Python je velmi vhodný pro širokou oblast použití, přestože ve výchozí instalaci neobsahuje nástroj pro tvorbu grafického uživatelského rozhraní. Python totiž dosahuje nejlepších výsledků ve většině sledovaných oblastí (tj. algoritmizace, procedurální model programování, objektový model programování, matematické aplikace, aplikace s textovým uživatelským rozhraním, programování pro web, práce se soubory, programování databázových aplikací, vykreslování bitmapové rastrové grafiky, vykreslování 3D grafiky, vytváření aplikací s grafickým uživatelským rozhraním, vhodnost syntaxe a účelnost vývojové prostředí). Přestože je Python primárně objektově orientovaný jazyk, je možné ho samozřejmě plnohodnotně využít i pro výuku na bázi procedurálního paradigmatu.

Autoři studie Why Complicate Things? Introducing Programming in High School Using Python (Mannila a kol., 2006) provedli výzkum na žácích střední školy, kteří absolvovali kurz programování v Pythonu. Výzkumu se účastnili studenti ve věku 16 až 19 let. Ti, kteří již s programováním měli zkušenosti, používali nejvíce jazyk Java nebo C++. Polovina z nich

řekla, že by pokračovali s programováním v Pythonu. Z výsledků výzkumu vyplynulo, že studenti vnímají Python jako jednodušší a zábavnější.

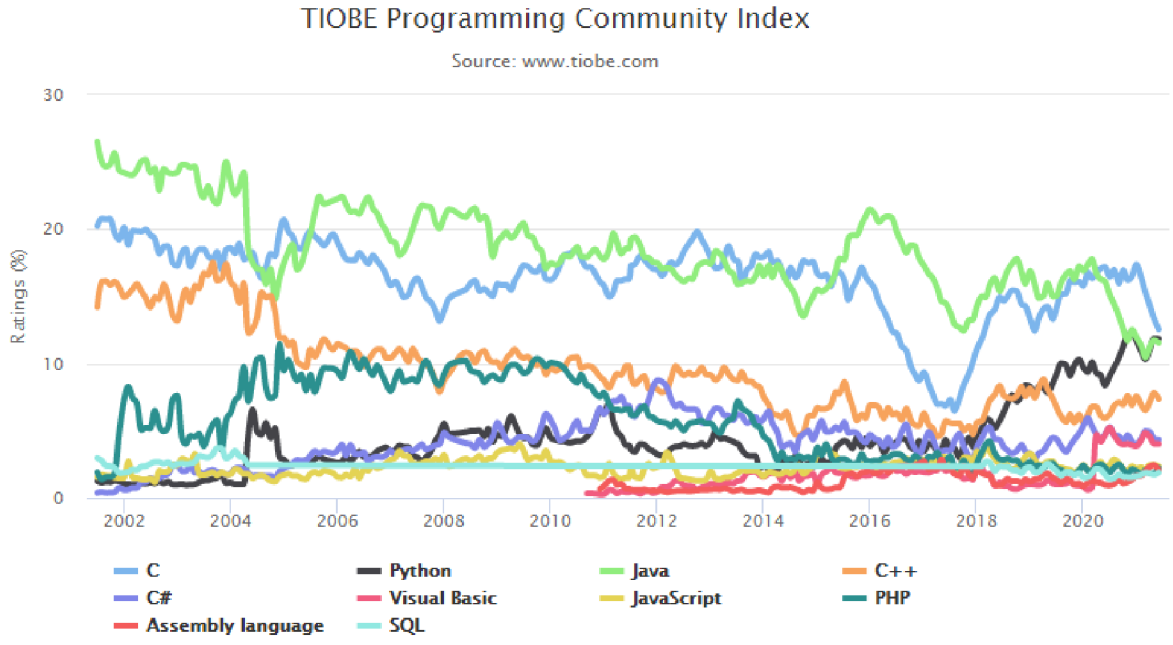
6.1 Python

Python je interpretovaný univerzální vysokoúrovňový programovací jazyk. Filozofie Pythonu zdůrazňuje čitelnost kódu díky jeho vynucenému využívání odsazování. Jeho konstrukce i objektově orientovaný přístup mají za cíl pomoct programátorům psát čistý a logický kód pro malé i velké projekty.

Python je dynamicky typovaný jazyk. Podporuje několik paradigmat, včetně strukturovaného (zejména procedurálního), objektově orientovaného a funkčního programování. Python je díky své obsáhlé standardní knihovně často popisován jako tzv. „batteries included“.

Aktuálnost a používanost

K posouzení aktuálnosti byl použit TIOBE Programming Community Index, jež je považován za relevantní ukazatel popularity programovacích jazyků. Z dlouhodobého hlediska vykazuje Python s jazykem C# mírný růst popularity, zatímco popularita Javy, C a C++ stále klesá. V roce 2021 se Python dokonce vyhoupl na stejnou pozici jako Java a C (viz Graf 3).



Graf 3: Dlouhodobý trend TIOBE Indexu pro prvních 10 programovacích jazyků

(Zdroj: https://www.tiobe.com)

Python využívají velké společnosti jako např. Seznam.cz, kde je spolu s C/C++ hlavním jazykem. Mezi další dobře známé společnosti využívající Python patří např. NASA, Red Hat, Panasonic, eBay a u nás ING, Národní knihovna atd.

Dostupnost

Python je tzv. open source software, takže je zdarma ke stažení na webových stránkách www.python.org pro různé operační systémy. Ve většině linuxových distribucí je také již předinstalován v základu. Pro psaní kódu si vystačíme s jakýmkoliv textovým editorem, např. poznámkovým blokem. Vhodnější je samozřejmě použít některé z moderních uživatelsky přívětivějších vývojových prostředí, jako např. Visual Studio Code, PyDev nebo IDLE.

Čitelnost kódu

Syntaxe Pythonu je jednoduchá a lehce naučitelná. Je inspirována jazykem ABC, který sloužil pro výuku začátečníků. Velkou výhodou je rychlost psaní kódu. Python může ušetřit až 80% času, např. oproti jazyku C. Napsat jednoduchý program je skutečně jednoduché. Např. C++ vyžaduje, aby byl program uvnitř funkce, Java zase, aby byl kód uvnitř třídy. Python si nic takového nevynucuje. Blok v Pythonu je definován odsazením a příkaz ukončen prostým koncem řádku. Tím odpadá klasický problém se zapomínáním závorek nebo středníků. Vynucené odsazování dělá kód čitelnější i pro jiné programátory. Počet řádků kódu navíc snižuje dynamické typování. Není potřeba deklarovat proměnné. Což naopak klade důraz na uvědomění si typu proměnné, se kterou pracujeme.

Rozšiřitelnost

Rozšíření jsou realizována pomocí modulů, které jsou analogií knihoven v jazyce C nebo Java. Modul je soubor funkcí, který můžeme naimportovat do našeho programu a přidat tím další užitečné funkce. Mezi základní knihovny patří např. math (matematické funkce), random (generátor náhodných čísel), datetime (práce s časem a datem) atd.

7 Specifikace požadavků na webovou aplikaci

Praktická část této práce má podobu průvodce tvorbou webové aplikace. Aplikace slouží výukovým účelům, přesněji rozvoji informatického myšlení. Cílem bylo vytvoření aplikace reálně využitelné např. v informaticky zaměřených kurzech v rámci volnočasových aktivit. Jako způsob, jakým podpořit rozvoj IM byla vybrána výuka základů programovacího jazyka Python, který je ve výuce hojně využíván. Nyní si specifikujeme požadavky na aplikaci, které definují celý proces vývoje:

- **přehlednost** – aplikace by měla být navržena podle tradičních schémat, která jsou ověřená praxí a podle standardů, pomáhajících používání co největšímu počtu uživatelů.
- **snadná orientace** – uživatel se musí snadno orientovat v prostředí webové stránky a dostat se na místo, které potřebuje, s co nejmenším počtem kliků. Špatně viditelné odkazy a jiné prvky mohou uživatele rychle odradit.
- **dobré grafické zpracování** – mělo by být vybráno takové barevné provedení, které souzní s věkem potenciálních uživatelů aplikace a zároveň nebude přehnaně rušivé. Přestože studie uvádějí např. vyvarování se velkých statických obrázků, u výukové aplikace pro děti bývají tyto pravidla benevolentnější.
- **kvalitní obsah** – u výukové aplikace platí více, než kdy jindy, že obsahová část je důležitější než forma. Je potřeba mít výukové texty ověřeny kvalitními zdroji tak, abychom uživatele za žádných okolností neuvedli v omyl.
- **bezpečnost** – není důvod, aby aplikace sbírala (ani vědomě) jakékoliv osobní informace od uživatelů a jakkoliv je zpracovávala. Systém se také musí umět vypořádat s nekorektními vstupy nebo přímo útoky od uživatele.
- **snadné ovládání** – ovládání musí být intuitivní i bez zbytečných doplňujících instrukcí. Nicméně u tohoto typu aplikace je vhodné texty doplnit o jasné a stručné vysvětlivky.
- **přístupnost** – pokud aplikace nenabízí personalizaci, není třeba využívat registrace a následného přihlašování. Tento přístup opět dokáže odradit velké množství uživatelů, stejně jako se stát potenciálním bezpečnostním rizikem. Přístupnosti napomáhá responzivní design, který dovoluje používání na rozmanitých typech zařízení.
- **bezplatnost** – tento bod souvisí s přístupností. Filozofií tohoto projektu je, že musí být přístupný všem bezplatně.
- **český jazyk** – jak jsme poznali z analýzy existujících nástrojů pro rozvoj IM, ryze českých aplikací je poskrovnu. Cílem je proto rozšíření možností výuky v ČR.

8 Metody a vývojové nástroje

Existují různé metody vedení projektu. Pro účely této práce byl vybrán klasický vodopádový model, který jsme si popsali v kapitole 5.4. Ten se dobře hodí pro menší projekty a jednočlenné týmy.

Důležitým bodem pro efektivní práci je výběr správných vývojových nástrojů. Ty si popíšeme v následujících odstavcích.

8.1 Editor kódu

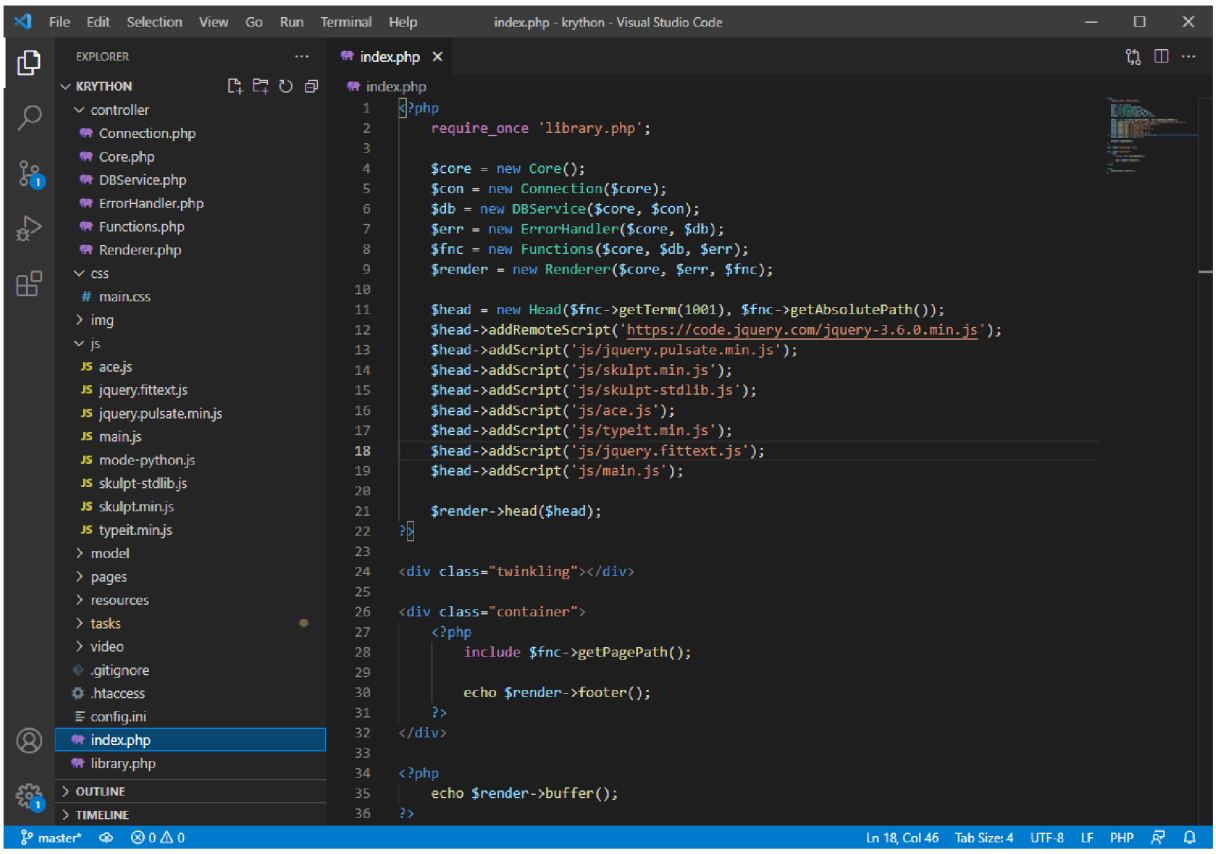
Ačkoliv by nám ke psaní kódu vystačil textový editor, moderní editory zdrojového kódu jsou schopné inteligentně rozpoznávat kód a asistovat při vývoji. Pro vývoj této aplikace byl vybrán produkt Visual Studio Code vytvořený společností Microsoft. Mezi hlavní funkce patří podpora ladění, inteligentní doplňování kódu, snippety, refaktorizace kódu a vestavěný verzovací systém Git. Uživatelé mohou měnit barevné schéma, klávesové zkratky, předvolby a instalovat rozšíření, která přidávají další funkce.

Microsoft uvolnil většinu zdrojového kódu editoru ve větvi microsoft/vscode na GitHubu pod licencí MIT. Všechny verze editoru jsou tedy freeware.

Ve výzkumu mezi uživateli na webu Stack Overflow z roku 2019 byl označen jako nejpopulárnější vývojové prostředí, když pro něj hlasovalo 50.7% z 87 317 respondentů.

Visual Studio Code lze použít s celou řadou programovacích jazyků, včetně Javy, JavaScriptu, Go, Pythonu, C++ aj. Tato základní podpora zahrnuje zvýrazňování syntaxe, párování závorek a skrývání kódu. Visual Studio Code také poskytuje IntelliSense pro JavaScript, TypeScript, JSON, CSS a HTML. Místo systému projektů umožňuje uživatelům otevřít jakýkoliv adresář a ten následně uložit do pracovního prostoru pro budoucí opakované použití (viz Obr. 17). To mu umožňuje fungovat jako editor kódu pro libovolný jazyk. Mnoho funkcí editoru není přístupných prostřednictvím nabídek nebo uživatelského rozhraní, ale lze k nim přistupovat prostřednictvím příkazové řádky.

Visual Studio Code lze obohatit o rozšíření dostupná prostřednictvím centrálního uložiště. To zahrnuje doplňky editoru a jazykovou lokalizaci. Pozoruhodnou vlastností je možnost vytvářet rozšíření, která přidávají podporu pro nové jazyky, motivy a ladicí programy.



Obr. 16: Prostředí Visual Studio Code

(Zdroj: vlastní snímek obrazovky)

8.2 Verzovací systém

V softwarovém inženýrství je správa verzí (také správa revizí) systémem odpovědným za správu změn počítačového kódu, dokumentů nebo jiných souborů informací. Jde v podstatě o uchovávání historie veškerých změn. Nejčastěji se používá pro kontrolu nad změnami zdrojového kódu při vývoji softwaru.

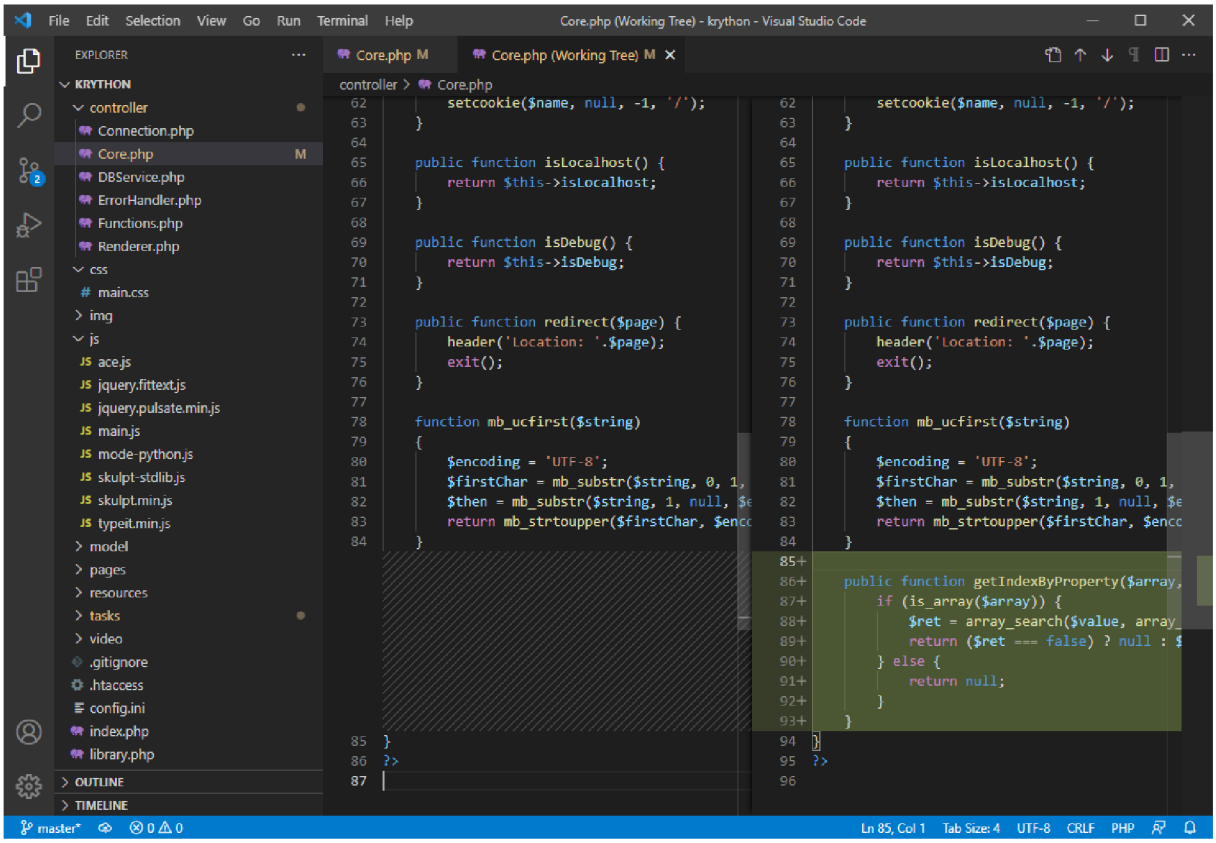
Systém správy verzí (Version control system, VCS) uchovává informace o tom, kdo, kdy a jak změnil které řádky kódu. Díky tomu lze procházet kompletní historii vývoje a v případě nežádoucího chování programu, se lze vrátit ke starší funkční verzi. Každé změně bývá přidělen identifikátor nazývaný revize. Nejznámějším systémem správy verzí je Git. Git byl původně vytvořen Linusem Torvaldsem pro vývoj jádra Linuxu a je open-source.

Významnou výhodou verzování je možnost spolupráce více programátorů na jednom projektu. Verzovací program totiž hlídá případné kolize a umí je dokonce automaticky řešit.

Dalším důležitým prvkem je ochrana proti kolapsu pracovní stanice. Z tohoto důvodu se obyčejně pro ukládání změn využívá vzdálený server. Data se ukládají do uložiště zvaného

repozitář. Pro potřeby tohoto projektu byla vybrána populární služba Bitbucket, která umožňuje vytvoření privátních repozitářů.

Systémy pro správu verzí běžně fungují jako samostatné aplikace, ale bývají také zabudované přímo ve vývojových prostředích. Visual Studio Code také dokáže pracovat s repozitářem Gitu a zálohovat jej na vzdálený server (viz Obr. 17).



Obr. 17: Systém pro správu verzí ve Visual Studio Code

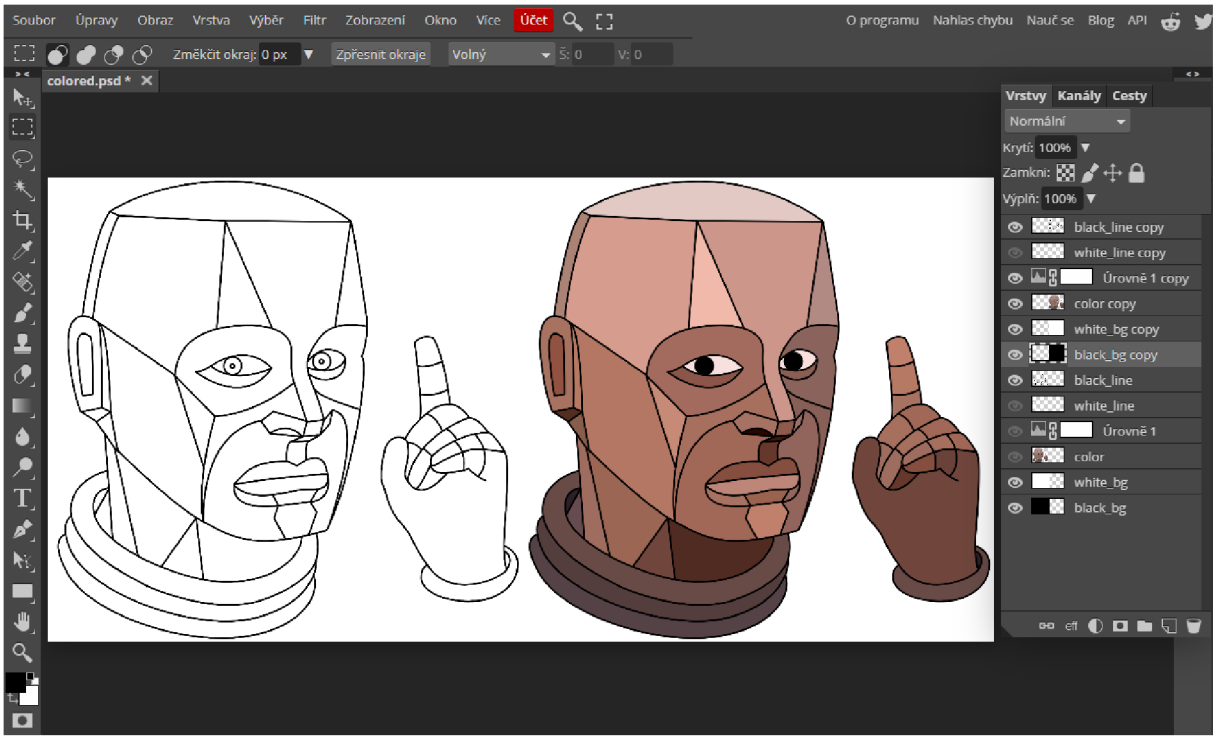
(Zdroj: vlastní snímek obrazovky)

8.3 Grafický editor

Aby bylo učiněno zadost autorským právům, je potřeba pro svou aplikaci samozřejmě vytvořit původní grafiku.

Zajímavým projektem je Photopea, online aplikace českého programátora s ukrajinskými kořeny Ivana Kutskira. Slouží k úpravě a zpracování rastrových i vektorových obrázků a je zdarma. Běží v prohlížeči, proto ji lze spustit na jakémkoliv zařízení bez nutnosti instalace. Jedná se o pokročilý editor grafiky, kompletně napsaný v JavaScriptu. Využívá WebGL, což je JavaScriptová grafická knihovna, vykonávaná přímo na grafické kartě, takže běh je plynulý. Má skoro totožné uživatelské prostředí jako placená aplikace Adobe Photoshop (viz Obr. 18), takže přesun z jedné aplikace do druhé je plynulý. Navíc je mnohem přívětivější

než např. konkurenční GIMP. Měsíčně aktuálně Photopea zaznamenává více než deset milionů otevření po celém světě.



Obr. 18: Tvorba grafiky v aplikaci Photopea

(Zdroj: vlastní snímek obrazovky)

8.4 HTTP Server

Pro testování a vývoj aplikace je potřeba mít připravené prostředí, kde aplikace poběží. Pro spuštění testovacího lokálního serveru byl použit balíček WampServer, což je aplikace, která na operačním systému Windows umí spustit Apache server s podporou PHP a MySQL databáze.

Pro vystavení produkční verze aplikace na internet je použita osobní doména www.vnovak.cz s vlastním webhostingem, také s podporou PHP a MySQL. Aplikace je tak dostupná z jakéhokoliv zařízení s přístupem k internetu. K nahrání souborů na FTP server je využívána open-source aplikace WinSCP.

9 Technologie použité při vývoji aplikace

Vzhledem k již existujícímu webhostingu a zkušenostech s těmito technologiemi se autor aplikace rozhodl použít tzv. schéma LAMP (viz Obr. 19), které sestává ze čtyř vrstev: Linux (operační systém), Apache (softwarový webový server), MySQL (databázový systém), PHP (programovací jazyk). Plus JavaScript (programovací jazyk interpretovaný na straně klienta), značkovací jazyk HTML a kaskádové styly CSS.

Obr. 19: Schéma LAMP

(Zdroj: https://codecondo.com/wp-content/uploads/2017/09/LAMP-Stack.jpg)

9.1 PHP

PHP je univerzální skriptovací jazyk vhodný pro vývoj dynamických webových aplikací. Původně jej vytvořil dánsko-kanadský programátor Rasmus Lerdorf v roce 1994. Nyní ho zaštiťuje společnost The PHP Group.

Kód je obvykle zpracováván na webovém serveru PHP interpreterem. Na webovém serveru se z výsledku provedeného kódu – což může být jakýkoliv typ dat, například vygenerovaná HTML stránka – utvoří část nebo celá HTTP odpověď. Syntaxe jazyka se inspiruje vícero programovacími jazyky (Java, Pascal, C a Perl). Jazyk PHP je nezávislý na platformě a skripty lze mezi operačními systémy přenášet bez modifikací.

PHP podporuje množství knihoven, např. pro práci se soubory, přístup k databázi (mj. MySQL, MSSQL, PostgreSQL, Oracle, ODBC), zpracování textu aj.

PHP je nejrozšířenějším jazykem pro tvorbu webových aplikací, k červenci 2019 měl podíl 79%. Jeho oblíbenost tkví v jednoduchosti a obrovské zásobě funkcí. Mezi nejznámější projekty napsané v jazyce PHP patří Wikipedie nebo Facebook. PHP je aktuálně ve verzi 8. V této verzi je napsána i aplikace tvořená v rámci této práce.

9.2 MySQL

MySQL je open-source relační databázový systém, vytvořený švédskou firmou MySQL AB, nyní vlastněný a vyvíjený společností Oracle.

Relační databáze organizuje data do jedné nebo více tabulek, ve kterých mohou být sloupce vzájemně propojeny; tyto vztahy pomáhají strukturovat data. Programátoři používají k vytváření, úpravám a extrahování dat z databáze jazyk SQL.

MySQL má samostatné klienty (např. phpMyAdmin), které uživatelům umožňují přímou interakci s databází MySQL pomocí SQL, ale především se používá při implementaci aplikací, které potřebují relační databázi. MySQL využívá mnoho populárních webových aplikací, jako např. Facebook, Flickr, YouTube, Twitter aj.

MySQL je napsán v C a C++. SQL parser je napsán v yacc. Funguje na mnoha systémových platformách, včetně Windows, macOS, Linux, FreeBSD, OpenBSD, OpenSolaris atd. Aktuálně existuje ve verzi 8, kterou využívá i naše aplikace. Aplikace však využívá databázi prakticky jen k načítání slovníku a chybových hlášek, žádný zápis záznamů není potřeba.

9.3 HTML

HTML (Hypertext Markup Language) je standardní značkovací jazyk navržený pro zobrazování dokumentů ve webovém prohlížeči.

Webové prohlížeče přijímají dokumenty HTML z webového serveru nebo z místního uložiště a vykreslují je na webové stránky. HTML popisuje strukturu webové stránky sémanticky a původně obsahoval i informace o vzhledu stránky.

Stavebními kameny stránek jsou HTML značky. Do vykreslené stránky mohou být vloženy obrázky, formuláře a další objekty. HTML poskytuje prostředky k vytvoření strukturovaných dokumentů pomocí množiny značek (tzv. tagů) pro nadpisy, odstavce, seznamy, odkazy, citace a další položky. Tagy často bývají párové. Názvy jednotlivých značek a jejich vlastností se uzavírají mezi úhlové závorky < a >.

V HTML může být vložen kód napsaný ve skriptovacím jazyce, jako je JavaScript, což ovlivňuje chování a obsah webových stránek. Zařazení CSS definuje vzhled a rozvržení obsahu. W3C (World Wide Web Consortium), bývalý správce HTML a současný správce standardů CSS doporučuje používání CSS pro definici vzhledu stránky. Od roku 2014 je HTML v nejnovější verzi 5.

9.4 CSS

Kaskádové styly (CSS) jsou jazyk pro popis způsobu zobrazení elementů dokumentu psaného v HTML, XHTML nebo XML. CSS umožňuje oddělení prezentace od obsahu, včetně rozložení, barev a písem. Toto oddělení zlepšuje přístupnost obsahu, větší flexibilitu a kontrolu při specifikaci vzhledu. Umožňuje více webovým stránkám sdílet formátování uložené v jednom samostatném souboru CSS, což sníží složitost a riziko opakování stejného kódu Uložení tohoto souboru v mezipaměti také zrychluje načítání stránek. Oddělení formátování a obsahu umožňuje prezentaci stejné stránky různým stylem podle způsobu vykreslení (obrazovka, tisk, audio forma nebo hmatový displej).

9.5 JavaScript

JavaScript je multiplatformní skriptovací jazyk, který odpovídá specifikaci ECMAScript. Svou syntaxí patří do rodiny jazyků C/C++/Java, ale sémantikou (funkcí) se zásadně liší. Jako multiparadigmatický jazyk podporuje funkční, procedurální a událostmi řízené programování. Má aplikační rozhraní pro práci s textem, daty, regulárními výrazy, standardními datovými strukturami a objektovým modelem dokumentu.

Vedle HTML a CSS je jednou ze stěžejních webových technologií. Používá jej přes 97% webových stránek. Všechny hlavní webové prohlížeče JavaScript umí interpretovat. Stejně jako CSS může být vložen přímo do HTML dokumentu nebo naimportován jako samostatný soubor.

JavaScript se spouští až po stažení stránky z Internetu (tedy na straně klienta). V dnešní době se však stále více začíná využívat i na straně serveru. Nejpoužívanější takovou implementací je Node.js.

9.5.1 jQuery

jQuery je JavaScriptová knihovna určená k lepší manipulaci s objekty dokumentu, k zjednodušení řízení událostí, CSS animací a Ajaxu. Poskytuje také vývojářům možnost vytvářet zásuvné moduly. Modulární přístup ke knihovně jQuery umožňuje vytvářet výkonné dynamické webové stránky a webové aplikace. jQuery je bezplatný open-source software s licencí MIT. V květnu 2019 jej používalo 73% z 10 milionů nejpoužívanějších stránek. Výzkumy ukazují, že jde o nejrozšířenější knihovnu, která má nejméně 3 až 4 krát větší využití než kterákoliv jiná JavaScriptová knihovna.

9.5.2 Skulpt

Další použitou knihovnou je Skulpt. Jedná se o implementaci Pythonu v JavaScriptu. Knihovna v zásadě dělá to, že dostane na vstupu kód Pythonu a vrátí výsledek. V případě chybného kódu vypíše konkrétní chybovou hlášku. Vše se děje na straně klienta, takže není vyžadována žádná součinnost serveru. Knihovna je také zdarma a dostupná z www.skulpt.org pod licencí MIT.

9.5.3 Ace

Aby bylo možné smysluplně využít možnosti knihovny Skulpt, potřebujeme editor kódu integrovaný v aplikaci, do kterého bude možné kód Pythonu vepsat. K tomu poslouží knihovna Ace. Ace je vestavěný editor kódu napsaný v JavaScriptu. Odpovídá funkcím a výkonu nativních editorů jako Sublime, Vim a TextMate. Lze jej snadno vložit do libovolné webové stránky a aplikace pro JavaScript. Zvládá syntaxi 110 programovacích jazyků (samozřejmě včetně Pythonu), automatické odsazení, dlouhý kód (až 4 milion řádků), klávesové zkratky, vyhledávání, zvýrazňování závorek, skryté znaky, zalamování řádků a skrývání kódu.

```
1 <div id="editor"><div>
2 <script>
3     var editor = ace.edit("editor");
4     editor.session.setMode("ace/mode/python");
5 </script>
```

Obr. 20: Příklad inicializace editoru Ace

(Zdroj: vlastní zpracování)

9.5.4 TypeIt

TypeIt je drobná univerzální knihovna, která umí vytvářet animace psacího stroje. Pro nekomerční užití je zdarma a v aplikaci je použita pro zobrazení výstupu kódu.

```
1 <p id="output"></p>
2 <script>
3     new TypeIt("#output", {
4         strings: "This is a simple string.",
5     }).go();
6 </script>
```

Obr. 21: Příklad inicializace animovaného textu TypeIt

(Zdroj: vlastní zpracování)

10 Implementace aplikace

Jelikož použití aplikace bylo zamýšleno převážně v informaticky zaměřených kurzech v rámci volnočasových aktivit, nejdůležitějším prvkem bylo, zprostředkovat výuku zábavnou formou.

Hlavní inspirací se stala webová aplikace SoloLearn, která ve výzkumu existujících aplikací pro podporu rozvoje informatického myšlení dopadla nejlépe. Cílem bylo implementovat, pokud možno, co nejvíce užitečných prvků z aplikace SoloLearn do aplikace tvořené v této práci a případně některé nevyhovující vynechat, či opravit. Bohužel není v časových možnostech vytvořit vícero kurzů pro různé programovací jazyky, proto byl jako nejvhodnější vybrán jeden jazyk, a to Python. Nicméně byl kurz navržen tak, aby důkladně seznámil uživatele se základními procedurálními konstrukty jazyka a to formou zábavných interaktivních úkolů. Pro celý kurz bylo vybráno jedno propojující téma.

10.1 Název a téma

Nejprve vznikl název aplikace, z kterého se vyvinulo téma protkávající jednotlivé úkoly. Python se anglicky vyslovuje ['paiθən]. Velmi podobně jako Kryten ['kɹaitən] v české překladu Kryton, fiktivní postava androida ze slavného britského sci-fi sitcomu Červený trpaslík. Tak přišel nápad na propojení těchto dvou světů a vznikl název aplikace (viz Obr. 22). Byla tedy vytvořena subdoména www.krython.vnovak.cz, přes kterou lze aplikaci spustit. Odkaz na aplikaci lze samozřejmě vyhledat i pomocí vyhledávače google.com.

Obr. 22: Logo aplikace Krython

(Zdroj: vlastní zpracování)

Červený trpaslík je nekonečnou inspirací pro vytváření zábavných interaktivních úkolů. Většina je originální, některé se však inspirují vtipnými gagy ze seriálu, či na ně jinak odkazují. Mají převážně formu algoritmického problému, s kterými se Kryton potýká a žádá uživatele o pomoc. Úkoly na sebe navíc částečně navazují a vytváří tak dojem celistvého příběhu.

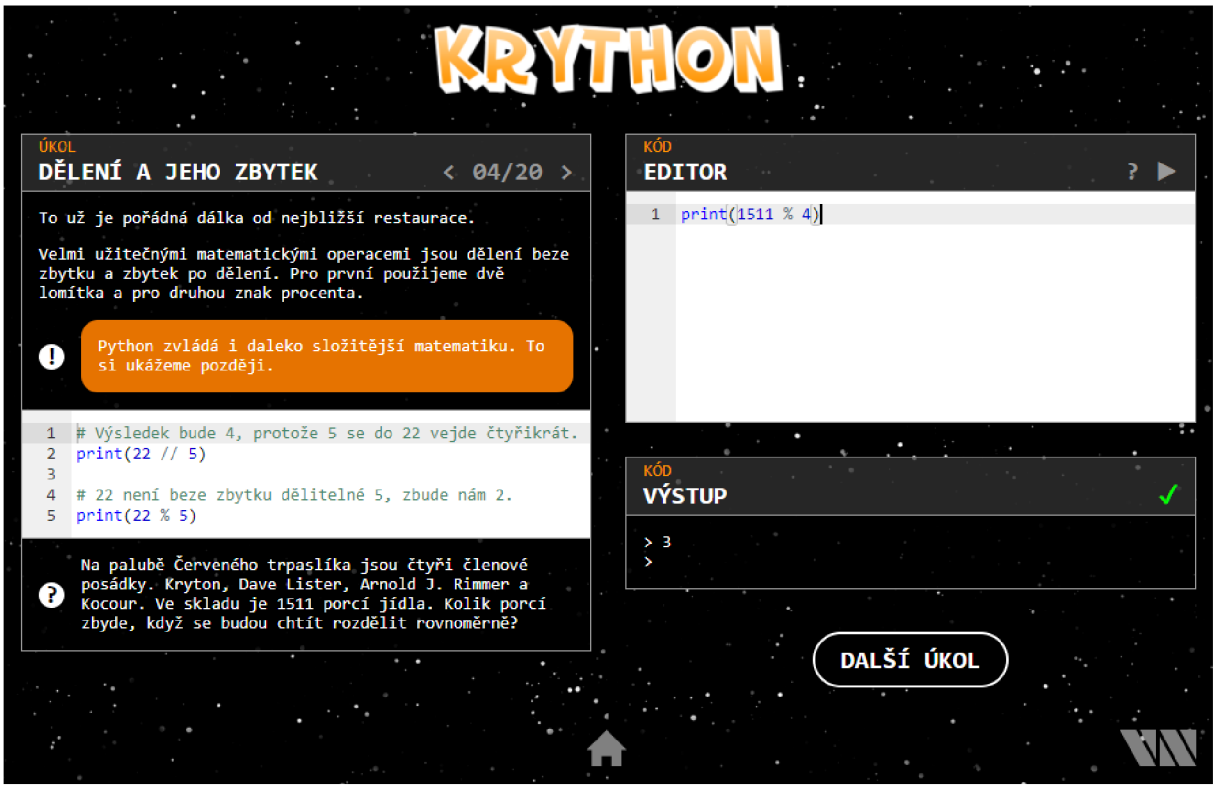
10.2 Prostředí aplikace

Po prvním spuštění do aplikace se zobrazí úvodní obrazovka s ilustrací Krytona a odkazy na jednotlivé úkoly (viz Obr. 23). Světlé čísla označují odemknuté úkoly. Tedy v tomto případě se uživatel může pokusit o splnění čtvrtého úkolu a odemknutí pátého. Počet odemknutých úkolů se ukládá do cookies prohlížeče. Takže když se uživatel vrátí k aplikaci příště, může pokračovat tam, kde skončil. Pokud však klikne na tlačítko „ZAČÍT ZNOVA“, počet odemknutých úkolů se vymaže. Aplikace byla od začátku navrhována tak, aby nevyžadovala registraci a následné přihlášení. Tato funkcionalita mnohdy odrazuje uživatele od používání aplikace, z důvodu vytváření nového účtu a pamatování si dalšího hesla. Předpokladem pro spuštění aplikace je samozřejmě přístup k internetu.

Obr. 23: Úvodní obrazovka aplikace Krython

(Zdroj: vlastní snímek obrazovky)

Prostředí jednotlivých úkolů si ukážeme na velmi jednoduchém čtvrtém úkolu, který učí matematické operace v Pythonu, dělení beze zbytku a zbytek po dělení (viz Obr. 24). Hlavní myšlenkou bylo, aby měl uživatel po ruce všechny části aplikaci i na relativně malém displeji. Jako konkrétně zde, kdy se vešly na, dnes již extrémně malou, obrazovku o rozlišení 1024x768 px. S přibývajícím textem se stránka samozřejmě stránka prodlužuje.



Obr. 24: Interaktivní úkol č. 4 aplikace Krython

(Zdroj: vlastní snímek obrazovky)

Grafické rozvržení stránky se vždy skládá ze tří bloků. Levý blok obsahuje (shora dolů):

- **název úkolu**
- **navigaci mezi úkoly (< 04/20 >)**
- **učební text**
- **dodatečné poznámky v oranžové bublině (!)**
- **needitovatelné ukázky kódu s vysvětlivkami (komentáři)**
- **zadání interaktivního příkladu (?)**

Zadání se s každým spuštěním aplikace drobně liší. Například zde se jedná o počet porcí jídla (1511). Aplikace se tak, do určité míry, snaží zabránit bezduchému kopírování kódu od jiných uživatelů.

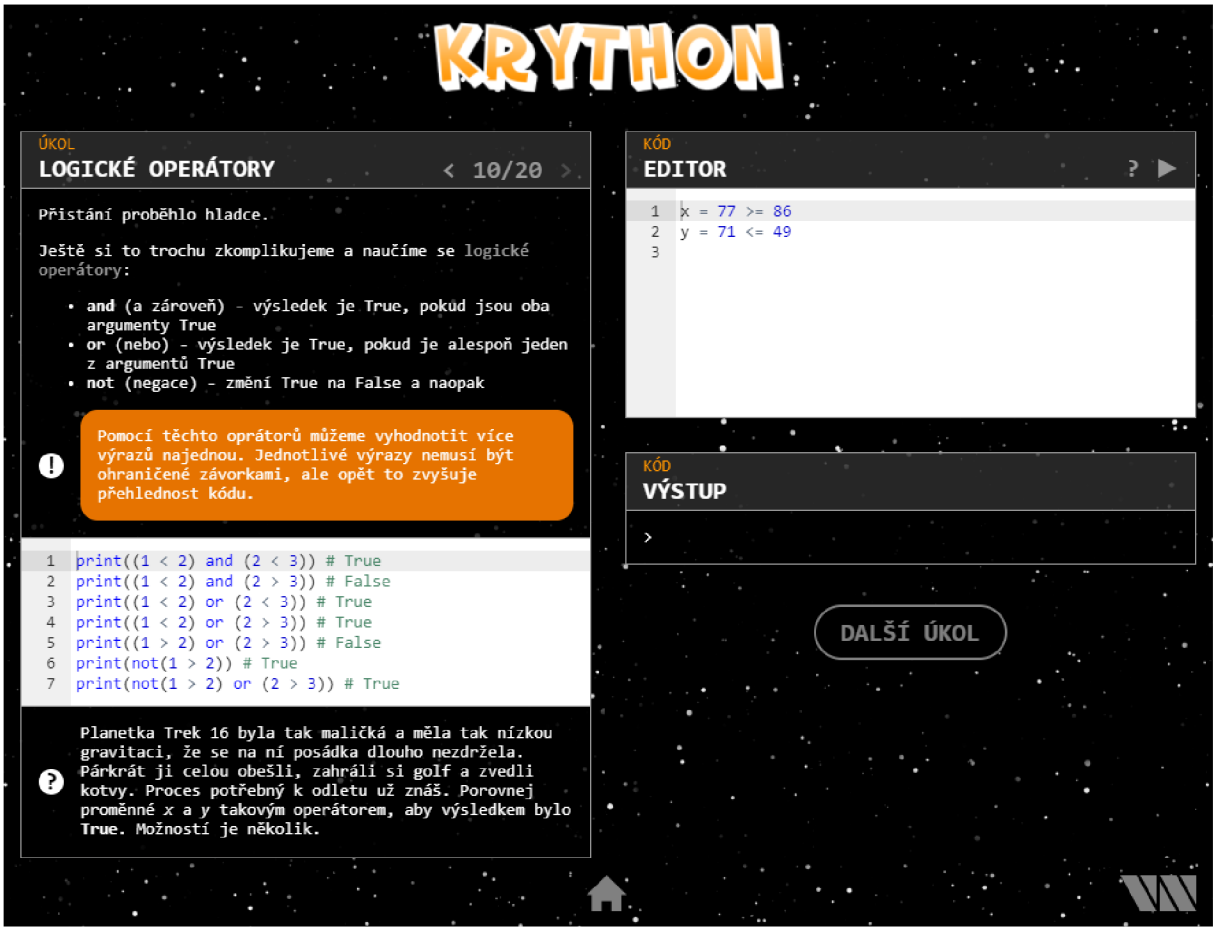
Na pravé straně se nachází „EDITOR“ (resp. konzole), do kterého uživatel vepisuje řešení úkolu a zároveň si může testovat chování programu. Editor využívá pro vykreslení JavaScriptovou knihovnu Ace. Blok obsahuje dva tlačítka. Tlačítko otazníku zobrazí uživateli nápovědu, tedy správnou podobu kódu, pokud si neví rady. Aplikace nemá sloužit jako soutěž, ani dovedností test, ale jako učební pomůcka. Proto stejně jako třeba SoloLearn, v případě bezradnosti, podá uživateli pomocnou ruku. Tlačítkem šipky (▶) se zadaný kód spustí a jeho

výsledek se zobrazí v bloku „VÝSTUP“ pod editorem. Pokud je výsledek správný, zobrazí se zelená fajfka, odemkne se další úkol a zpřístupní tlačítko „DALŠÍ ÚKOL“. V opačném případě se zobrazí červený křížek a chybová hláška.

Na úvodní obrazovku se uživatel dostane kliknutím na ikonku domečku ve spodní části. Podobně podrobný popis celého prostředí a fungování aplikace samozřejmě také najdeme v prvním úkolu, aby se uživatel byl schopný ihned sám orientovat.

Trochu složitější případ představuje úkol 10 (viz Obr. 25), kdy už jsou v editoru napevno přednastavené proměnné, uživatel je nemůže měnit a musí s nimi pracovat takovým způsobem, aby získal správný výsledek.

V tomto úkolu jsou navíc použity hypertextové odkazy na externí zdroje informací, pro přesnější vysvětlení určitých pojmů. Konkrétně zde odkaz „logické operátory“ vedoucí na stránku Wikipedie.



Obr. 25: Interaktivní úkol č. 10 aplikace Krython

(Zdroj: vlastní snímek obrazovky)

Responzivní design

Responzivní design je přístup k webovému designu, díky kterému jsou webové stránky dobře vykreslovány na různých zařízeních a velikostech obrazovek. Nejedná se však pouze o vlastnost prvků stránky měnit velikost, ale rovněž přeskupovat se podle velikosti obrazovky. Ideálně tak, aby uživatel na mobilním zařízení vůbec nemusel scrollovat horizontálně, ale pouze vertikálně, tak jak bývá zvykem. Doporučuje se však aplikaci používat primárně na PC s klávesnicí, aby bylo psaní kódu co nejkomfortnější. Responzivitu ukazují obrázky 26 a 27.

Obr. 26: Responzivní design úvodní obrazovky aplikace Krython

(Zdroj: vlastní snímek obrazovky mobilního zařízení)

Obr. 27: Responzivní design interaktivního úkolu č. 4 aplikace Krython

(Zdroj: vlastní snímek obrazovky mobilního zařízení)

K dosažení správné responzivity nebylo potřeba využívat dnes velmi oblíbený framework Bootstrap. Ten je pro většinu projektů zbytečně robustní. Zkušený programátor si bohatě vystačí s moderními vlastnostmi jazyka CSS3, určenými pro efektivní pozicování položek na stránce, souhrnně nazývanými flexbox. Pomocí flexboxu lze inteligentně zarovnávat a rozdělit prostor mezi položkami, a to i v případech, kdy je jejich velikost neznámá nebo dynamická. Hlavní myšlenkou je takové uspořádání, aby co nejlépe zaplnilo volné místo a zároveň nepřetékalo nikam mimo obrazovku.

10.3 Obsah interaktivních úkolů

Obsah úkolů byl pečlivě vybrán tak, aby dostatečně pokryl základy programování v jazyce Python. Podklady pro výběr obsahu aplikace byly knihy, např. Ponořme se do Python(u) 3 od Marka Pilgrima (Pilgrim, 2010) nebo Think Python: How to Think Like a Computer Scientist od Allena Downeyho (Downey, 2015), učebnice a vzdělávací materiály ze stránek www.imysleni.cz nebo konkurenční výukové aplikace. Teorie je uživateli podávána stručnou formou, aby nenudila a neodvracela pozornost od hlavní myšlenky: učení hrou.

Použité výukové metody:

- **Klasické metody**
  - **slovní** – práce s textovým materiálem
  - **názorně-demonstrační** – ukázky kódu
  - **dovednostně praktická** – experimentování
- **Aktivizující metody**
  - **výzkumná** – zkoumání, hledání řešení
  - **heuristická** – řešení problému (algoritmu)
- **Komplexní metody**
  - **individuální výuka** – samostatná práce
  - **výuka podporovaná počítačem**

Popis jednotlivých úkolů

1. Výpis textu

První úkol především představuje uživateli prostředí aplikace a učí nejzákladnější konstrukt jazyka, kterým je výpis textu na obrazovku – příkaz print. Uživatel se naučí základní datový typ string. Vytvoří si svůj první program a zjistí, jak funguje editor kódu a výpis do konzole.

2. Základní matematické operace

Obsahem jsou základní matematické operace jako sčítání, odčítání, násobení a dělení a rozdíl mezi datovým typem integer a float.

3. Umocňování

V dalším kratičkém úkolu je představena jiná užitečná matematická operace, a to umocňování, v Pythonu realizované příkazem **. Ke slovu se dostává tzv. priorita výrazů. Je vysvětleno, že umocňování má vyšší prioritu než předchozí matematické operace.

4. Dělení a jeho zbytek

Jako poslední ze základních matematických operací je představeno dělení beze zbytku a zbytek po dělení. Připomínána je možnost použití jak datového typu integer, tak float.

5. Přetypování

Po představení tří základních datových typů a operací s nimi, přichází na řadu přetypování, což je převod z jednoho datového typu na jiný. Funkce pro přetypování jsou int(), float() a str().

6. Proměnné

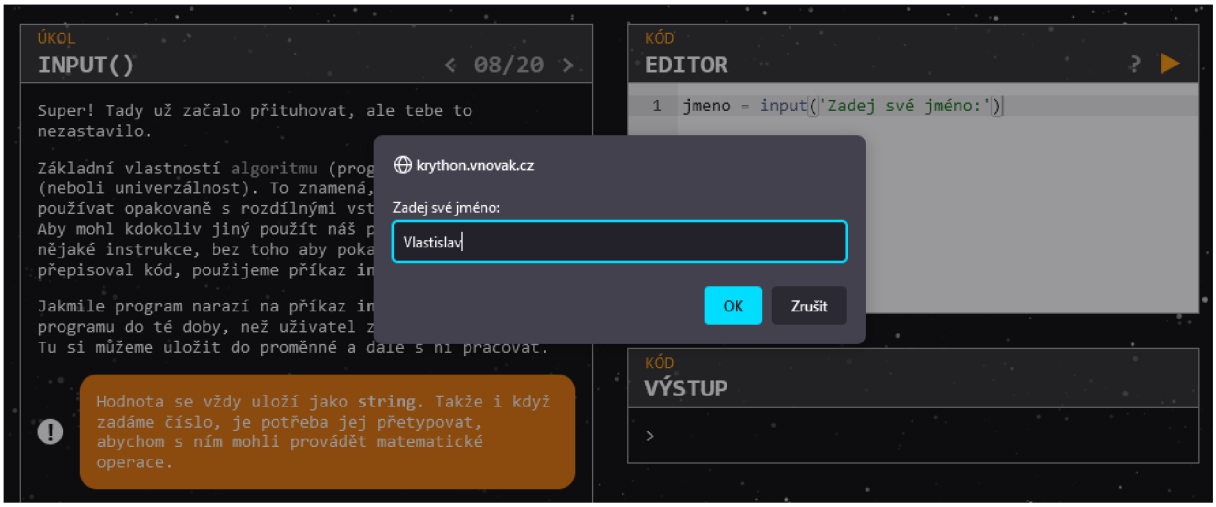
V tomto úkolu jsou vysvětleny proměnné a jejich význam, jakožto nepostradatelné části jakéhokoliv programovacího jazyka. Je popsáno jakým způsobem je můžeme pojmenovávat a jakých hodnot mohou nabývat. V příkladu je poprvé použita technika, kdy v editoru jsou napevno předdefinované náhodné proměnné a uživatel je musí využít k tomu, aby se dopracoval ke správnému výsledku.

7. Moduly

Možná trochu překvapivě jsou už v této fázi představeny moduly. Většina ostatních studijních materiálů je vysvětluje daleko později. Zde jsou však zásadní pro zvýšení interaktivity nadcházejících úkolů – např. použití knihovny random pro náhodné plnění vstupních proměnných. Nadto jsou vysvětleny velmi triviálně a jejich pochopení by nemělo dělat uživateli problémy.

8. Uživatelský vstup

Další základní funkcí je tzv. uživatelský vstup, realizovaný funkcí input. Ten dovoluje tzv. znovupoužitelnost (neboli univerzálnost) programu. Po spuštění kódu, který obsahuje tuto funkci, se uživateli zobrazí dialogové okno, kde může zadat jakoukoliv hodnotu (viz Obr. 28).



Obr. 28: Ukázka dialogového okna v aplikaci Krython

(Zdroj: vlastní snímek obrazovky)

9. Porovnávací operátory

Tento úkol již vyžaduje větší zapojení logického uvažování, jelikož se zabývá datovým typem boolean a porovnávacími operátory (`==`, `>`, `<`, `>=`, `<=`, `!=`).

10. Logické operátory

K porovnávacím operátorům z předchozího úkolu si přidáme logické operátory (and, or, not). Zdůrazněna je funkce závorek při vyhodnocování více výrazů najednou.

11. Podmínky

Dalším zásadním konstruktem jazyka, který uživateli dovolí vytvářet složitější programy, jsou podmínky. Dostatečně je vysvětlena důležitost odsazování a jeho standardní užití. S tímto úkolem se postupně zvětšuje délka ukázkových kódů a náročnost příkladů, díky zvyšující se komplexnosti kódu.

12. Seznamy

Tento úkol otevírá sadu tří po sobě jdoucích úkolů, věnujících se seznamům (neboli listům). Tato problematika je natolik komplexní, že si žádá více času na vysvětlení. Zde se popisuje tvorba seznamů (pomocí inicializace a funkce range) a vybírání prvků. Je zde také ukázáno proč se string chová podobně jako seznam.

13. Operace se seznamy

Tato část představuje další operace se seznamy, jako je změna prvku, sčítání a násobení listů a ověření, zda se prvek nachází v seznamu.

14. Funkce seznamů

Poslední část týkající se seznamů popisuje funkce pro přidání prvku (append, insert), odstranění prvku (remove, pop), zjištění počtu prvků (len) a na jaké pozici se prvek nachází (index).

15. Slovníky

Podobným datovým typem jako seznamy jsou slovníky. S tím rozdílem, že každý prvek slovníku obsahuje klíč a hodnotu. Funkce, které používají slovníky, jsou taky odlišné. Kromě známé funkce pop pro odstranění prvku, jsou vysvětleny nové funkce keys a values.

16. Cyklus for

V této části je představen cyklus for jako nejvhodnější pro iteraci přes různé sekvence (seznam, slovník, string) a je popsána funkce příkazů break a continue.

17. Cyklus while

Cyklus while naopak iteruje tak dlouho, dokud je splněna nějaká podmínka. I zde lze použít příkazy break a continue. Je potřeba dávat pozor obzvlášť na to, aby cyklus neběžel do nekonečna. V takovém případě dojde k zahlcení aplikace a jejímu pádu.

18. Funkce

Další základní konstrukcí, kterou je vhodné zde zařadit, je funkce. Jsou zde popsány hlavně různé formy předávání argumentů funkci (žádný argument, pojmenovaný argument, výchozí hodnota argumentu, seznam nebo slovník jako argument).

19. Funkce řetězců

Tento úkol ukazuje pár užitečných funkcí využitelných při práci s řetězci. Patří mezi ně formátování textů, spojení listu do řetězce a naopak, nahrazení části řetězce jiným, převod na velká, malá písmena atd.

20. Výjimky

V předchozích úkolech jistě každý narazil na spoustu různých logických chyb při zkoušení kódu. V mnoha případech tyto chyby nemusí jít na vrub programátora. V takové situaci je lze ošetřit pomocí tzv. výjimek, tak aby program nespadl a mohl pokračovat dál. Přestože je vždy žádoucí předcházet chybám jinými způsoby, někdy jsou výjimky jediná možnost, jak zajistit stabilitu programu. A tento poslední úkol učí jak na to.

Závěr

Cílem práce bylo vytvoření výukové webové aplikace pro podporu rozvoje informatického myšlení. Aplikace je nyní vytvořena v předpokládané kvalitě a rozsahu. Je plně funkční a dostupná zdarma komukoliv z internetové adresy www.krython.vnovak.cz. Cíl práce byl tímto naplněn.

Autor má v plánu aplikaci předložit učitelům informatiky prostřednictvím jemu dostupných kontaktů. Např. skupina uživatelů na sociální síti Facebook s názvem „Učíme informatiku“ má v současnosti dva tisíce členů. Je spravována Jednotou školských informatiků (www.jsi.cz) a její členové jsou velmi sdílní a vděční za každou výpomoc či nové studijní materiály.

Autor také počítá s budoucími opravami, či aktualizacemi. Aplikace obsahuje kontakt na autora pro případné dotazy či připomínky.

Seznam použitých zdrojů

Literatura

ANGELI, Charoula a Valanides NICOS. *Developing young children's computational thinking with educational robotics: An interaction effect between gender and scaffolding strategy.* Computers in Human Behavior, 2020. DOI: 10.1016/j.chb.2019.03.018. ISSN 07475632.

BALANSKAT Anja, Katja ENGELHARDT a Hanna Alexandra LICHT. *Strategies to include computational thinking in school curricula in Norway and Sweden - European Schoolnet's 2018 Study Visit.* European Schoolnet, Brussels, 2018.

BECK, Kent. *Extrémní programování*. Grada Publishing, 2002. 160 s. ISBN 80-247-0300-9,

BEDNAŘÍKOVÁ, Iveta. *Kapitoly z andragogiky* 2. Vyd. 1. Olomouc: Vydavatelství UP, 2006. 82 s. ISBN 80-244-1193-8.

BENDL, Stanislav. *Vychovatelství: učebnice teoretických základů oboru.* Praha: Grada, 2015. 312 s. ISBN 978-80-247-4248-9.

BOCCONI, Stefania, Augusto CHIOCCARIELLO, Giuliana DETTORI, Anusca FERRARI, a Katja ENGELHARDT. *Developing computational thinking in compulsory education – Implications for policy and practice*, 2016. DOI: 10.2791/792158.

BRENNAN, Karen a Mitchel RESICK. *New frameworks for studying and assessing the development of computational thinking*. In: Proceedings of the 2012 Annual Meeting of the American Educational Research Association, Vancouver, Canada, 2012.

DEHNADI, Saeed. *A cognitive study of early learning of programming.* London: School of Engineering and Information Sciences Hendon, 2009.

DOWNEY, Allen. *Think Python: How to Think Like a Computer Scientist.* Massachusetts: Green Tea Press, 2015. 222 s. ISBN: 978-1449330729.

DRAGON, Tomáš a Milan KLEMENT. *Comparison of Approaches to the Use of Web and Mobile Applications in Teaching Algorithmization and Programming in the Czech Republic and Abroad*. In: 12th annual International Conference on Education and New Learning Technologies (s. 6926-6935). Madrid: IATED Academy, 2020. ISBN: 978-84-09-17979-4.

DRAGON, Tomáš. *Support of Teacher's Work in the Field of Development of Computational Thinking Through E-Learning Resources.* In: The 3rd International Conference on Education and Multimedia Technology. New York: ACM, 2019, s. 131–135. ISBN: 978-1-4503-7210-7.

ĎURÁKOVÁ, Daniela, Jiří DVORSKÝ a Eliška OCHODKOVÁ. *Základy algoritmizace.* Ostrava: VŠB Technická univerzita Ostrava, Katedra informatiky, 2002.

Evropská komise. *Opening up education: Innovative teaching and learning for all through new technologies and open educational resources.* Brussels: Commision of Europian Communities, 2013.

HÁJEK, Bedřich, Břetislav HOFBAUER a Jiřina PÁVKOVÁ. *Pedagogické ovlivňování volného času: současné trendy.* Praha: Portál, 2008. 240 s. ISBN 978-80-7367-473-1.

HEŘMANOVÁ, Jana a Milan MACEK. *Metodika pro podporu tvorby školního vzdělávacího programu ve školských zařízeních pro zájmové vzdělávání.* Praha: Ministerstvo školství, mládeže a tělovýchovy, Odbor pro mládež, 2009. 110 s. ISBN 978-80-86784-77-9.

HOFBAUER, Břetislav. *Děti, mládež a volný čas.* Praha: Portál, 2004. 176 s. ISBN 80-7178-927-5.

CHEN, Guanhua, Ji SHEN, Laure BARTH-COHEN a Shian JIANG. *Assessing elementary students' computational thinking in everyday reasoning and robotics programming.* Computers & Education, 2017, 109(7), s. 162-175. DOI:10.1016/j.compedu.2017.03.001.

KAPLÁNEK, Michal. *Volný čas a jeho význam ve výchově.* Praha: Portál, 2017. 208 s. ISBN 978-80-262-1250-8.

KAPLÁNEK, Michal. *Metodiky zájmových činností 1.* České Budějovice: Jihočeská univerzita. Teologická fakulta, 2006. Studijní materiál. Nepublikovaný rukopis.

KLEMENT, Milan. *Traditional topics for the framework educational programme focused on ICT area, and the perception of these topics by the primary school ninth grade pupils.* Journal of Technology and Information Education, 2018, 10(1), s. 43-62. ISSN 1803-537X.

KLEMENT, Milan, Tomáš DRAGON a Lucie BRYNDOVÁ. *Computational Thinking and How to Develop it in the Educational Process.* 1. vyd. Olomouc: Palacký University Olomouc, 2020. 218 s. ISBN 978-80-244-5797-0.

KRATOCHVÍLOVÁ, Emília. *Pedagogika voľného času: výchova v čase mimo vyučovania v pedagogickej teórii a v praxi.* 1. vyd. Bratislava: Vydavateľstvo UK, 2004. 307 s. ISBN 80-223-1930-9.

NAS – National Academies of Science. *Report of a Workshop on The Scope and Nature of Computational Thinking*. Washington, D.C.: National Academies Press, 2010. DOI: 10.17226/12840. ISBN 978-0-309-14957-0.

NĚMEC, Jiří. *Kapitoly ze sociální pedagogiky a pedagogiky volného času: pro doplňující pedagogické studium.* Brno: Paido, 2002. 119 s. ISBN 80-7315-012-3.

PAPERT, Seymour. *Mindstorms: children, computers, and powerful ideas.* New York, USA: Basic Books Inc., 1980. 238 s. ISBN 978-0-85527-163-3.

PÁVKOVÁ, Jiřina. *Volný čas – pole působnosti vychovatele.* In: BENDL, Stanislav a kol. Vychovatelství: učebnice teoretických základů oboru. Praha: Grada, 2015. s. 118-156. ISBN 978-80-247-4248-9.

PÁVKOVÁ, Jiřina. *Pedagogika volného času.* Praha: Univerzita Karlova, Pedagogická fakulta, 2014. 145 s. ISBN 978-80-7290-666-6.

PÁVKOVÁ, Jiřina a kol. *Pedagogika volného času: [teorie, praxe a perspektivy výchovy mimo vyučování a zařízení volného času].* Vyd. 4. Praha: Portál, 2008. 221 s. ISBN 978-80-7367-423-6.

PILGRIM, Mark. *Ponořme se do Python(u) 3.* Praha: CZ.NIC, 2010. 430 s. ISBN: 978-80-904248-2-1.

ŠERÁK, Michal. *Zájmové vzdělávání dospělých.* Praha: Portál, 2009. 208 s. ISBN 978-80-7367-551-6.

TRAN, Yune. *Computational Thinking Equity in Elementary Classrooms: What Third-Grade Students Know and Can Do*. Journal of Educational Computing Research, 2017, 57(1), s. 3-31. DOI: 10.1177/0735633117743918.

VÁŽANSKÝ, Mojmír. *Základy pedagogiky volného času.* 2. upr. a dopl. vyd. Brno: Print-Typia, 2001. 175 s. ISBN 80-86384-00-4.

VOOGT, Joke a kol. *Computational thinking in compulsory education: towards an agenda for research and practice*. Education and Information Technologies, 2015, 20(4), s. 715-728. DOI: 10.1007/s10639-015-9412-6.

VYMAZAL, Jiří. *Mimoškolská výchova a vzdělávání dospělých a její institucionální systém v ČSR*. Praha: Státní pedagogické nakladatelství, 1990. 140 s. ISBN 80-7066-104-6.

WING, Jeannette M. *Computational thinking*. Communications of the ACM, 2006, 49(3), s. 33. DOI: 10.1145/1118178.1118215.

Elektronické zdroje

ATWOOD, Jeff. *Separating Programming Sheep from Non-Programming Goats*. Coding Horror: programming and human factor. Berkeley, 2006. [cit. 2021-05-28]. Dostupné z: https://blog.codinghorror.com/separating-programming-sheep-from-non-programming-goats/

BÍLÁ KNIHA – národní program rozvoje vzdělávání v ČR. In: *MŠMT*, 2001. [cit. 2021-05-14]. Dostupné z: https://www.msmt.cz/file/35405_1_1/.

BRET, Victor. *Learnable Programming: Designing a programming system for understanding programs*. Worry Dream. Berkeley, 2012. [cit. 2021-05-28]. Dostupné z: http://worrydream.com/LearnableProgramming/.

CSTA and ISTE. *Operational Definition of Computational Thinking for K-12 Education, 2011*. [cit. 2021-05-27]. Dostupné z: https://cdn.iste.org/www-root/ct-documents/computational-thinking-operational-definition-flyer.pdf.

ČESKO. Vyhláška č. 74 ze dne 31. července 2018 o zájmovém vzdělávání. In: *Sbírka zákonů České republiky*, 2018, částka 84, s. 2586-2591. [cit. 2021-05-14]. Dostupné z: http://www.msmt.cz/file/48248_1_1/.

HEGGART, Keith. *Coded for Success: The Benefits of Learning to Program*. Edutopia. George Lucas Educational Foundation. Marin County (California), 2014. [cit. 2021-05-30]. Dostupné z: http://www.edutopia.org/discussion/coded-success-benefits-learning-program.

KOTEK, Lukáš. *Výzkum používaných programovacích jazyků na středních školách*. Praha, 2013. [cit. 2021-05-30]. Dostupné z: https://dspace5.zcu.cz/bitstream/11025/16677/1/kotek_vyzkum_clanek.pdf.

MANNILA, Linda, Mia PELTOMÄKI, Ralph-Johan BACK a Tapio SALAKOSKI. *Why Complicate Things? Introducing Programming in High School Using Python.* 2006. [cit. 2021-05-28]. Dostupné z: https://www.researchgate.net/publication/31596786_Why_Complicate_Things_Introducing_Programming_in_High_School_Using_Python.

PECINOVSKÝ, Rudolf. *Jak efektivně učit OOP*. Příspěvek na konferenci Tvorba softwaru 2005, Ostrava, 2005. [cit. 2021-05-28]. Dostupné z: http://www.vyuka.pecinovsky.cz/prispevky/2005-SW_Jak_efektivne_ucit_OOP.pdf.

PITNER, Tomáš. *Výuka programování na základní a střední škole*, 2000. [cit. 2021-05-28]. Dostupné z: https://www.fi.muni.cz/~tomp/semuc/text_pitner.doc.

PRENSKY, Marc. *Digital Natives, Digital Immigrants Part 1*. On the Horizon, 2001, 9(5), s. 1-6. DOI: 10.1108/10748120110424816. ISSN 1074-8121. [cit. 2021-05-27]. Dostupné z: https://www.marcprensky.com/writing/Prensky%20-%20Digital%20Natives,%20Digital%20Immigrants%20-%20Part1.pdf.

Střediska volného času. In: *MŠMT*, 2013- 2021. [cit. 2021-05-19]. Dostupné z: https://www.msmt.cz/mladez/strediska-volneho-casu.

THE ROYAL SOCIETY. *Shutdown or restart?: The way forward for computing in UK schools. Education Section.* London, 2012. [cit. 2021-05-27]. Dostupné z: https://royalsociety.org/-/media/education/computing-in-schools/2012-01-12-computing-in-schools.pdf.

VANÍČEK, Jiří. *Výuka algoritmizace patří především do informatiky.* Příspěvek na konferenci Počítač ve škole 2016 v Novém Městě na Moravě. České Budějovice: Jihočeská univerzita, 2016. [cit. 2021-05-28]. Dostupné z: http://www.pocitacveskole.cz/system/files/soubory/sbornik/2016/vanicek1.pdf.

WING, Jeannette M. *Computational thinking benefit society.* Social Issues in Computing blog, 2014. [cit. 2021-05-27]. Dostupné z: http://socialissues.cs.toronto.edu/index.html%3Fp=279.html.

Zájmové vzdělávání. In: *MŠMT*, 2013- 2021. [cit. 2021-05-19]. Dostupné z: https://www.msmt.cz/mladez/zajmove-vzdelavani-1.

Kvalifikační práce

DOHNAL, Pavel. *Programování na základních školách*. Brno, 2009. Diplomová práce. Masarykova univerzita, Pedagogická fakulta. Vedoucí práce Ing. Martin Dosedla.

RÝDLO, Lukáš. *Programovací jazyky pro výuku programování na SŠ*. Brno, 2012. Diplomová práce. Masarykova univerzita. Fakulta informatiky. Vedoucí práce doc. RNDr. Tomáš Pitner, Ph.D.

ŠIŠKOVÁ, Kateřina. *Volný čas dětí, mládeže a elektronická média*. Olomouc, 2010. Diplomová práce. Univerzita Palackého, Filozofická fakulta, Katedra žurnalistiky. Vedoucí práce Mgr. Renáta Sedláková, Ph.D.

Seznam obrázků

Seznam grafů

Seznam tabulek

Anotace

Jméno a příjmení:	Bc. Vlastislav Novák
Katedra:	Katedra technické a informační výchovy
Vedoucí práce:	Mgr. Tomáš Dragon
Rok obhajoby:	2021

Název práce:	Tvorba webové aplikace pro podporu rozvoje informatického myšlení
Název v angličtině:	Development of web application to support the development of computational thinking
Anotace práce:	Práce se zabývá tvorbou výukové webové aplikace pro účastníky počítačových kurzů v rámci volnočasových aktivit. Úkolem aplikace je rozvoj informatického myšlení osvojením si základních algoritmů a jejich konstrukcí, reprezentovaných programovacím jazykem Python. Teoretická část představuje termín informatické myšlení a hodnotí existující výukové aplikace zaměřené na algoritmizaci a výuku programování. Praktická část se soustředí na samotný vývoj webové aplikace pomocí jazyka PHP a dalších webových technologií.
Klíčová slova:	informatické myšlení, algoritmizace, programování, výuka, Python, webová aplikace
Anotace v angličtině:	The aim of the theses is to create educational web application for IT courses within leisure activities. Target of the application is development of computational thinking by adopting basic algorithms and their design, represented by the programming language Python. The theoretical section introduces the term computational thinking and evaluates existing educational applications in field of algorithms and programming. The practical part focuses on the actual development of the web application using the PHP language and other web technologies.
Klíčová slova v angličtině:	computational thinking, algorithm, programming, education, Python, web application
Přílohy vázané v práci:	-
Rozsah práce:	89 stran
Jazyk práce:	čeština