UNIVERZITA PALACKÉHO V OLOMOUCI

PEDAGOGICKÁ FAKULTA

Katedra technické a informační výchovy

Diplomová práce

Bc. Ladislav Orel

Učitelství technické a informační výchovy pro střední školy
a 2. stupeň základních škol a učitelství geografie pro střední školy.

Programování dříve a dnes, změny přístupu ve výuce programování v ČR

Olomouc 2018

Vedoucí práce: doc. PhDr. Milan Klement, Ph.D.

Čestné prohlášení

Svým podpisem zde stvrzuji, že diplomovou práci: *„Programování dříve a dnes, změny přístupu ve výuce programování v ČR“* jsem vypracoval samostatně. Veškeré použité materiály a podklady, ze kterých byly čerpány informace, jsou uvedeny v seznamu použité literatury a jsou řádně ocitovány dle citační normy ČSN ISO 690.

V Olomouci dne 9. 4. 2018 …….

Ladislav Orel

PODĚKOVÁNÍ

Vzhledem k neveselým okolnostem, které mě postihly při zpracovávání původní diplomové práce, která musela být kvůli jistým okolnostem přerušena, patří poděkování hlavně panu docentu Milanu Klementovi, který mě v mé svízelné situaci přijal, moji kvalifikační práci vedl a poskytoval cenné rady. V druhé řadě pak všem, kteří mě v době mého studia, jakkoliv podporovali.

Obsah

Úvod

Dnešní svět patří moderním technologiím, nicméně i v dobách minulých bylo zapotřebí využit tehdejších *„moderních“* technologií a ty stále zlepšovat a bořit hranice možností.

Kdysi dávno, když se zrodil na Zemi život, se dal do pohybu celkový proces pokroku. Člověk se vyvinul z primátů a začal používat jednoduché nástroje, později začal vytvářet nástroje z kovu. Lidé z Číny dali světu papír, Římané vytvářeli stabilní stavby pomocí římského betonu, lidé začali tisknout knihy, zkrotili těžkou techniku v době průmyslové revoluce, lidé začali štěpit atom… Všechny tyto věci a vynálezy dali lidstvu další možnosti a posunuly nás vpřed. Tyto možnosti vždy vycházely z předešlých nabytých vědomostí a zkušeností. Aby mohli lidé pokračovat dál, nesměli se nových technologií bát a museli se je postupně naučit používat v každodenním životě.

Dnešní doba, doba informační a digitální, patří informacím, jejích získávání, filtrování, zpracovávání, ukládání, reprodukování. Informace se dnes šíří pomocí informačních a komunikačních technologií. Jejich užití v našem dnešním světě je naprostou nutností a celý svět je na koloběhu informací zcela závislý. Je proto nutné umět tyto technologie používat. Součástí informačních a komunikačních technologií je ovšem nejen jejich užívání, ale také jejich vytváření a neustálé vylepšování. Tak jako lidé minulosti své vynálezy vylepšovali, tak i lidé přítomnosti nesmí polevit a zastavit se na jednom místě.

V dnešním světě je vytváření prostředků a možností na poli informačních a komunikačních technologií nazýváno *„programováním“.* Programování umožňuje *„oživit chladný stroj“*, dát mu úlohu a nechat stroj pracovat, bez větší nutnosti tento stroj neustále kontrolovat (i když to je také občas potřeba). Stroj pak výpočetním výkonem poskytuje možnost urychlení vědeckého pokroku a celkového vývoje společnosti – možnosti vytvářet dokonalejší vymoženosti budoucnosti již *„dnes“*. Programování ale není jednoduchou činností, vyžaduje obrovské úsilí, snahu, čas a trpělivost. Ne každý člověk může být dobrým programátorem, naopak každý se ale může naučit aspoň jednoduchou formu programování, a tím celkově přispět k částečnému pokroku. Tato práce má za cíl poukázat na nutnost seznámení člověka s činností programování co nejdříve. Tak, jako se dříve lidé museli naučit jisté věci každodenního života, tak by se dnes měli lidé dostávat k programování. Celý vývoj lidské společnosti se neustále urychluje, proto by se lidé na programování neměli dívat, jako na něco složitého, čí hanlivého a měli by mu jít naproti.

„Každý člověk v této zemi by se měl naučit programovat, protože ho to naučí, jak myslet.“

Steve Jobs (zakladatel firmy Apple)

Hlavní cíl práce

Kvalifikační práce má za cíl čtenářovi objasnit význam programování a poukázat na to, proč je (v dnešní době) programování důležité. V práci je nastíněna letmá historie programování, jak vlastně programování vzniklo, kterým směrem se následně ubíralo a práce je dále doplněna o letmou prognózu, jak by mohlo programování za pár let vypadat. Po historii je čtenář seznámen se způsoby, jakým je výuka programování a programování samotné realizováno. Znázorněna je i aktuální situace na pracovním trhu v oblasti ICT a programování. Samotná práce neobsahuje pouze teoretickou část, nýbrž i část empirickou, kde jsme zjišťovali, jak na tom ČR v oblasti výuky programování byla, jakým způsobem výuka programování probíhá nyní a kam se nejspíše bude ubírat dále. Celá empirická část práce je následně popsána, okomentována a doplněna o grafické prvky, tabulky a obrazové materiály.

1. Historie ve světě programování

1.1. Vymezení pojmu – Programování

Za programování označujeme v informatice proces, kterým je možné navrhnout řešení určitého problému pomocí výpočetní techniky. Cílem programování je vytvoření spustitelného počítačového programu, který po spuštění zadaný problém vyřeší. Programování zahrnuje analýzu problému, pochopení problému, vytvoření algoritmu, a nakonec vhodný zápis do zdrojového kódu v příslušném programovacím jazyce (What is Coding?, 2014). Ne vždy je ale práce hned napoprvé hotová správně. U problému, ke kterému se hledá řešení, může existovat více způsobu, jak dosáhnout tohoto řešení – programy proto mohou mít více způsobů, jak ke správnému výsledku dojít. Optimalizace je pak proces, kterým se snažíme původní program vylepšit tak, aby jeho nová verze byla efektivnější, nebo snížila nároky na výpočetní výkon (Wilson, 1983).

Vymezení pojmu programování s historií jako takovou přímo nesouvisí, nicméně bylo třeba samotné programování pro potřeby kvalifikační práce definovat co nejdříve.

1.2. První programovatelný počítač

Pro programování je zapotřebí počítače. Za první programovatelný počítač je možné považovat diferenční stroj, který mechanicky počítal hodnoty polynomických funkcí. Tento stroj byl navrhnut Charlesem Babbagem, který ho však nesestrojil kompletně. To se stalo až v roce 1991 (120 let po jeho úmrtí), kdy byl podle jeho původních plánů sestaven diferenční počítač pouze za pomocí součástek a prostředků dostupných v 19. století. I přesto, že tedy on sám počítač nesestrojil (pouze navrhnul), je považován za *„otce počítačů“* (The modern history of computing, 2006). I když diferenční stroj na první pohled působil dost netaktně, jeho základní struktura se podobala dnešním počítačům, které mají oddělenou programovou a datovou paměť. Operace probíhaly pomocí instrukcí, bylo možné provádět skoky v programu a vstupní a výstupní jednotky byly oddělené (Gleick, 2011).

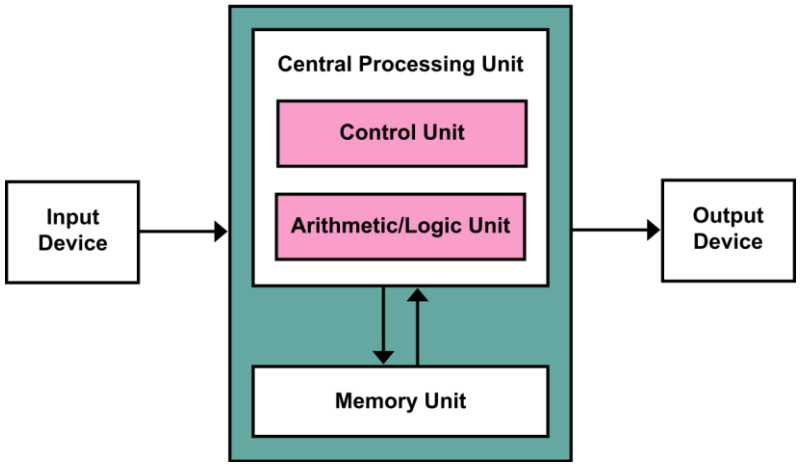
Počítače se původně programovaly pomocí děrovacích štítků (např. právě zmíněný diferenční Babbagův stroj), kdy paměťové medium bylo reprezentováno tenkým kartonem a informace v něm byly zapsány formou dírek na určité pozici. Tato metoda zápisu však byla dost zdlouhavá, náročná a na chyby se většinou přišlo až při počítání samotným strojem. Když stroj zahlásil chybu, štítek byl přeposlán zpět na opravu (George, 2006). Prvním programátorem, který navrhl programování tak, jak ho známé dnes, byla Augusta Ada King, která detailně popsala popis fungování Babbagova diferenčního stroje, zavedla pojmy: podmíněný skok, nepodmíněný skok, cyklus a podprogram. Mimo samotný popis obsahovaly její poznámky také algoritmus pro výpočet Bernoulliho čísel, stroj však v době vzniku algoritmu ještě neexistoval fyzicky, a tak nebyl program nikdy otestován (Parker, 2014).

Počítače nadále fungovaly na stejném principu, co se týče programování, vyvíjely se však v oblasti technické. Babbagův diferenční stroj byl poháněn ručním pohonem, později měl hnací sílu dodávat parní stroj a dále byly stroje přizpůsobeny tomu, aby je mohl pohánět elektrický proud. Významným strojem byl pak počítač ENIAC, jenž je historicky prvním elektronkovým počítačem (Shurkin, 1996).

1.3. JOHN VON NEUMANN A JEHO PŘÍNOS

John von Neumann byl významný rakousko-uherský matematik, který velice přispěl k mnoha vědním oborům, mezi nimi i informatice. Roku 1945 navrhl koncept *„jednoduchého“* počítače, který říkal, že: *„počítač by měl být univerzálním strojem, který by neměl být předem stavěn pro cílový program, ale naopak, aby cílové programy byl stavěny pro jeden „typ“ počítače.“* (Ferguson, 2000).

Výsledkem tedy bylo, že program byl uložen do paměti počítače společně se zpracovávanými daty. Dále se mohlo zpětně vracet k určitým částem výpočtů pomocí cyklů či podmínek. Program se tedy dal opakovaně používat bez nutnosti předchozího zdlouhavého nahrávání a celkově se tak urychlil celý chod stroje (Ferguson, 2000).



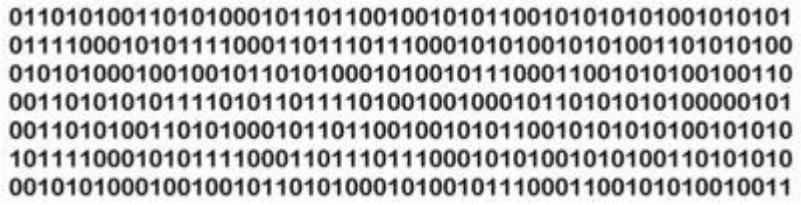
Obrázek 1: Schéma počítače podle John Von Neumanna. Zdroj: Wikipedia.org

1.4. KOMPILÁTOR

Stále rychleji se vyvíjející technologie si žádaly i urychlení programování. Stroje, ač byly tehdy velké a robustní (ENIAC byl 10 metrů vysoký, 100 metrů dlouhý a 3 metry hluboký), tak výpočetní operace prováděly kvalitně a rychle v rámci hodin (lehčí pak i v řádech minut). Lidé, kteří ale stroje programovali, svoje programy leckdy připravovali dny, někdy i týdny (Weik, 1955).

Programovalo se ve strojovém kódu, který je reprezentován sekvencí bitů, jelikož stroj byl schopen přijmout informace jen v této podobě. Programování ve strojovém kódu má 2 velké nevýhody. První nevýhodou je velice náročné skládání programu, kde se vyskytuje veliká pravděpodobnost vzniku chyby zapříčiněné lidským faktorem. Druhá velká nevýhoda spočívá

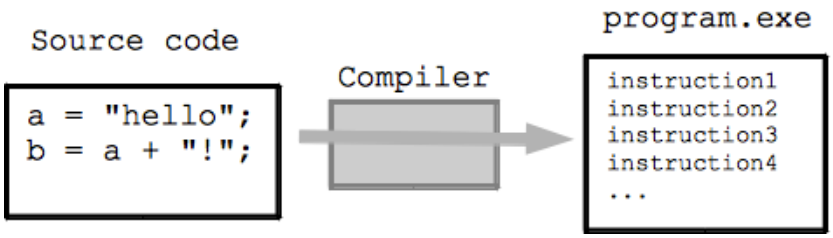
v abstrakci – pokud se člověk neoprostí od pouhých jedniček a nul, nemůže tvořit úkony na základě *„skutečných“* počítačových operací a tak není schopen vytvořit důmyslnější, lepší a výhodnější řešení v kratším časovém úseku (Freedman, 2011).



Obrázek 2: Ukázka binárního kódu

Pojmem kompilátor[1] je v oblasti informatiky označen software, který je uzpůsoben k tomu, aby přeložil programové instrukce z *„lidské“* řeči do strojového kódu (PCMag, 2017).

V roce 1951 byl vyvinut první jednoduchý kompilátor A-0. Vytvořila ho Grace Hooperová[2], americká matematička a počítačová vědkyně. Jeho funkce spočívala v zavádění instrukcí a převádění jejích adres přímo do strojového kódu. Překlad do strojového kódu (zde pouze činnost dat a jejich adresy) pak značně urychlil proces samotného programování, navíc se zamezilo vzniku chyb vzniklé lidským faktorem (Ferguson, 2000).



Obrázek 3: Ukázka práce kompilátoru. Zdroj: introcomputing.org

1.5. LIDŠTĚJŠÍ FORMA PŘÍKAZŮ – PROGRAMOVÝ JAZYK

Programování ve strojovém kódu bylo zdlouhavé a složité, programování v lidštější formě by pro programátora bylo jednodušší, rychlejší a efektivnější. Aby však mohlo probíhat, musel vzniknout nástroj, který by jednoduše vysvětlil stroji, co má dělat. Programovací jazyk pak byl tímto nástrojem, kterým programátor sděloval stroji, co se má vykonat (Běhálek, nedat.).

1.6. VYŠŠÍ A NIŽŠÍ PROGRAMOVACÍ JAZYK

Programovací jazyky můžeme rozdělit na vyšší programovací jazyky a nižší programovací jazyky. Vyšší programovací jazyky jsou pak logicky nad nižšími. Vyšší zde značí to, že jazyk je snazší na pochopení – co do vzhledu i formy (příkladem může být heslovitě popsána instrukce). Nižší

[1] z anglického *to compile* = sestavit, vytvořit

[2] Byla zaměstnaná u námořnictva spojených států. Počítače byly hojně využívány pro výpočet trajektorie balistických střel (WWII). Armáda si vědoma intelektu Grace Hooperové ji zaměstnala jako programátorku pro tehdejší počítače.

programovací jazyk je pak převedená forma instrukcí s nízkou úrovní nebo zcela žádnou abstrakcí. Výsledkem může být to, že i dobrý programátor vyšších programovacích jazyků se v kódu nevyzná, protože je popsán v nižším programovacím jazyku (Educba, 2015).

Java Code	Asembler Code

`import java.util.Scanner;`
 	
`public class Hello`
`{`
`  public static void main(String[] args)`
`  {`
`  System.out.print("Please enter your name: ");`
`  String name = new Scanner(System.in).next();`
 	
`  System.out.println("Hi "+name+"!");`
`  }`
`}`
	`foo:`
	`movl $0xFF001122, %eax`
	`addl %ecx, %edx`
	`xorl %esi, %esi`
	`pushl %ebx`
	`movl 4(%esp), %ebx`
	`leal (%eax,%ecx,2), %esi`
	`cmpl %eax, %ebx`
	`jnae foo`

Obrázek 4: Rozdíl mezi vyšším a nižším programovacím jazykem

1.7. ASEMBLER

Přesněji tedy Assembly Language, jazyk symbolických adres (nesprávně označován jako Asembler, nicméně tento název *„zdomácněl“ a* běžně se nadále používal) byl nižší programovací jazyk. Vznikl v roce 1949 ve firmě IBM a jeho hlavní funkce spočívala v tom, že si programátor nemusel pamatovat číselné kódy jednotlivých instrukcí, dále odpadla potřeba výpočtu adres, skoků a umístění dat. Celkově se celý zápis programu zjednodušil a zkrátil. Nutno podotknout, že Assembly Language byl do strojového kódu překládán překladačem *„Assembler“*, který byl mylně zaměňován se samostatným programovacím jazykem Assembly Language, respektive jeho zdomácnělým pojmenováním Asembler. Později byl nahrazen vyššími programovacími jazyky s vyšší úrovní abstrakce a širším využitím skrze kreativitu a produktivitu programátorů (Assembly language, 1990).

1.8. FORTRAN

Jedním z prvních vyšších programovacích jazyků se stal Fortran[3]. Jazyk vznikl v roce 1957 taktéž zásluhou firmy IBM. Vytvořen byl hlavně pro výpočty a numerické aplikace. Obsahoval příkazy GOTO, IF, DO a logické proměnné (Ferguson, 2000). Dnes má stále své uplatnění např. pro výpočty trajektorii raket, Fourierův rozvoj, Fourierovu transformaci, výpočty vývoje počasí, elektroinženýrství nebo částicovou fyziku. Je také hojně využíván pro programy, které zatěžují a testující super počítače (ACMQUEUE, 2010).

Celkový vývoj Fortranu k dokonalosti trval 18 let, a i poté docházelo k řadě implementací (např. doplnění o instrukce: THEN, ELSE, IF). Všechny novější verze byly s původní kompatibilní (nebo fungovala možnost překladu). Programovací jazyk byl tak oblíbený a rozšířený, že existovalo

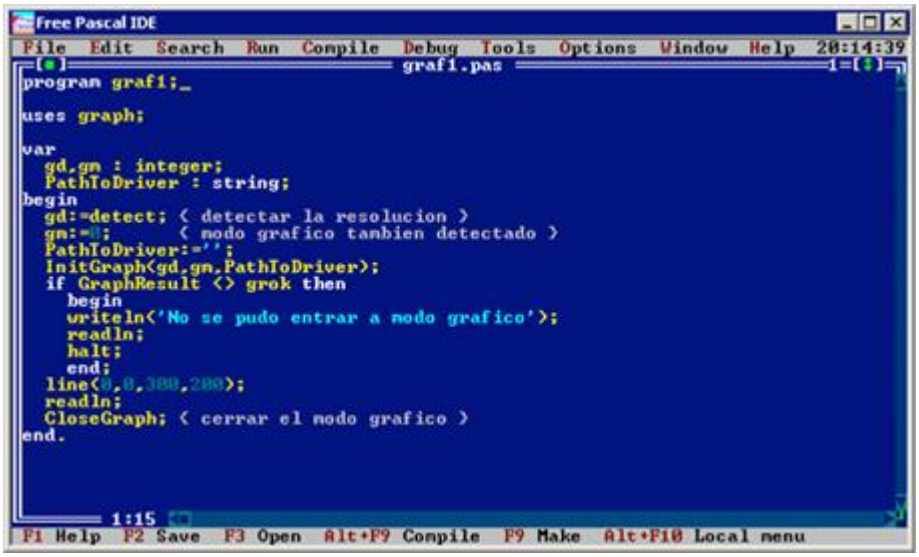
[3] Fortran se skládal z 2 anglických slov – FORmulta TRANslator = volně přeloženo jako překladač vzorců.

mnoho kompilátorů pro různé počítačové architektury. Hlavní výhodou programovacího jazyka byla práce s čísly a numerickými operacemi, práce s textem byla o něco obtížnější (Wexelblat, 1981). Zprvu byli lidé vůči vyšším programovacím jazykům velice skeptičtí a nechtěli je využívat. Také nevěřili, že je možné pomoci nich vytvořit jednodušší formu strojového kódu, než např. v Asembleru. Kompilátor nakonec ale vytvořil kód, který byl úhlednější než ručně psaný kód v Asembleru. Tyto výhody mu pak vynesly statut velice oblíbeného a používaného nástroje k programování. Sám autor John Backus řekl v interview pro časopis Think, že jeho práce vzešla z toho, že byl líny psát programy, neměl to rád, a aby si usnadnil práci, začal pracovat na programovacím systému, který by mu práci ulehčil (NBC, 2017).

1.9. ÉRA VYŠŠÍCH PROGRAMOVACÍCH JAZYKŮ

Po masivním rozšířením Fortranu se začaly rozšiřovat v hojné míře i další vyšší programovací jazyky. Programovací jazyk ALGOL[4] vznikl v roce 1958. Byl využíván hlavně pro standardní metodu popisu algoritmů, zavedl kód v blocích a párování pomocí BEGIN a END pro ohraničení jednotlivých bloků. ALGOL se stal předlohou pro další programovací jazyky jako např. Pascal a C (Backus, 1960).

Jak již bylo řečeno, jazyk Pascal se inspiroval programovacím jazykem ALGOL. Vznikl v roce 1970, jeho autorem byl Niklaus Wirth a vznikl hlavně pro potřeby výuky programování. Cílem bylo vytvořit *„jednoduchý“* a hardwarově nenáročný programovací jazyk vhodný pro většinu tehdejších počítačů. Jazyk byl založen na omezeném počtu srozumitelných konstrukcí, i přes to bylo mimo výuku programování možné v programovacím jazyce vytvořit reálné a užitečné programy (Ferguson, 2000).



Obrázek 5: Prostředí TurboPascalu, nástavba jazyka Pascal. Zdroj: bbvaopen4u.com

[4] z anglických slov: ALGOritmic Language = algoritmický jazyk

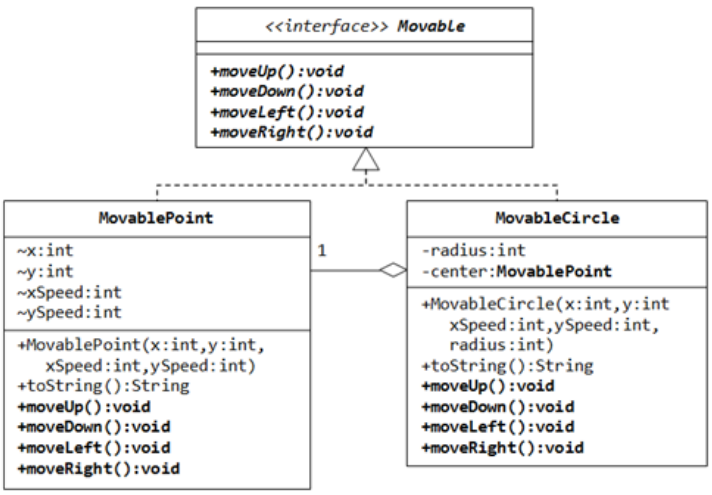
Další z programovacích jazyků (relativní – vzhledem k Assembleru vyšší, vzhledem k Javě nižší programovací jazyk), je jazyk C, který vznikl v roce 1972 a ještě dnes je stále oblíbený a hojně se využívá. Programovací jazyk byl vyvinut Kenem Thompsonem a Dennisem Ritchiem pro potřeby operačního systému UNIX[5]. Jazyk je vhodný k programování různého druhů programů, ale nejčastěji je užíván pro vytváření systémového softwaru. Jeho výhodou je, že je čitelnější než Assembler, ale umožňuje jeho zápis přímo do zdrojového kódu a je přenositelný i na jiné procesory a jiné architektury. Jeho nevýhodou však je *„ukazatel“*, což je způsob odkazování v paměti a odkazování na funkce. Program je nefunkční nebo nestabilní, pokud programátor odkáže na nesprávné umístění (Ferguson, 2000).

1.10. OBJEKTOVĚ ORIENTOVANÉ PROGRAMOVÁNÍ

Přelom 70. a 80. let přináší specifický způsob programovacího stylu – filosofie a způsob designu programů. Programování je uskutečňováno pomocí objektů, které mezi sebou navzájem komunikují. Pokud se někde v programu objeví chyba, zpravidla stačí opravit jenom tu jistou část (Úvod do objektově orientovaného programování v C#, 2012).

Objektově orientované programování (dále jen OOP) se opírá o 3 hlavní základní pilíře: zapouzdření, dědičnost a polymorfismus. Zapouzdření lze vysvětlit jako uzavřený systém, ke kterému lze přistupovat pouze skrze určité rozhraní – nehrozí pak možnost neautorizovaného přístupu zvenčí do objektu. Dědičnost lze využít v případě znovupoužití vybraných objektů, či vytvoření nových na základě dříve použitých/starých. Polymorfismus pak označuje možnost používat stejné rozhraní pro různé typy objektů (Úvod do objektově orientovaného programování v C#, 2012).

Základem OOP je modelovat situace reálného světa, oprostit se od toho, jak vnímá kód počítač a zaměřit se na pohled ze strany programátora (popř. následného uživatele). Mezi známé OOP jazyky můžeme jmenovat např. C++ (rozšíření jazyka C), Java, C#, PHP a mnoho jiných (Dědičnost a polymorfismus, 2012).



Obrázek 6: Zapouzdřené objekty v programovacím jazyku Java. Zdroj: Nanyang Technology University, Singapore

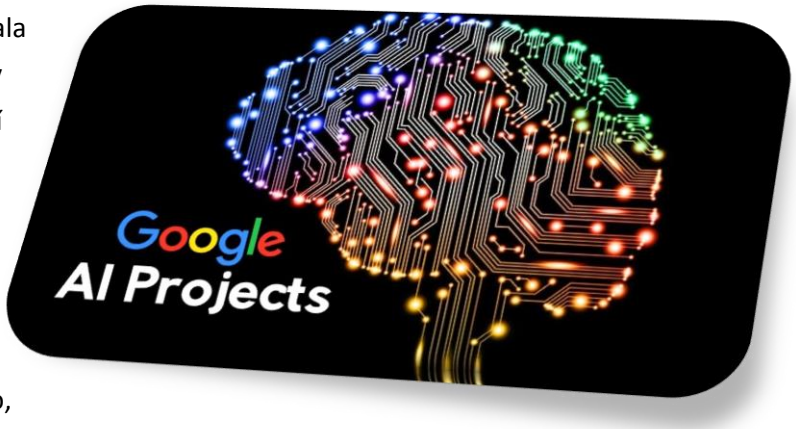
[5] Operační systém vytvořený v Bellových laboratořích firmy AT&T a také výraz pro *„otevřený operační systém“*

1.11. NÁHLED DO BUDOUCNOSTI

Dříve bylo programování náročným úkonem. Hlavním cílem bylo zprostředkovat jednoduchý program pomocí jednoduchých instrukcí – hlavně proto, že tehdejší počítače byly výkonnostně oproti dnešním značně pozadu. Dnes v podstatě není problém ve výkonu počítače, ale spíše zaznamenat miliardy instrukcí do jednoho funkčního programového komplexu ve správném znění. Od textového psaní se ale v nejbližší době přesuneme k tomu, že počítači sdělíme úkol a počítač se poté sám postará o vytvoření kódu (Tanz, 2016).

Další, již ne moc vzdálenou, budoucností je pojem AI[6] – umělá inteligence a její užití v programování. Samo o sobě je téma umělé inteligence dost náročné. Je to specifická oblast informatiky, která se zabývá tvorbou strojů vykazujících známky inteligentního chování. Zde bude uveden jenom záměr jejího použití v programování, a to sice kvůli schopnosti umělé inteligence zlepšovat svůj vlastní program (dalo by se říci zlepšovat sama sebe) učením se z předešlých úkonů. Společnost Google vybrala jedny z nejlepších programátorů světa, aby pracovali na projektu umělé inteligence, která sama vytvoří jinou – dokonalejší formu umělé inteligence. V tomto případě umělá inteligence vytváří novou formu sebe sama tím, že jsou eliminovány chyby lidského faktoru. Program pohání počítač, který je schopen *„vymyslet“* více možných způsobů za kratší čas, vybrat ten nejlepší a jak již bylo řečeno, *„učit se z minulosti“*. Programátoři z Googlu uspěli, umělá inteligence vytvořená lidmi a následná umělá inteligence vytvořená samotnou umělou inteligencí měla za úkol pouze rozeznávat v obrazových souborech jednotlivé elementy a ty určitým způsobem nějak charakterizovat/“zaškatulkovat“ (lidé, tvary, zvířata apod.). Zatímco původní umělá inteligence vytvořená lidmi měla úspěšnost 39 %, úspěšnost nové umělé inteligence činila 43 %, čímž bylo dokázáno, že došlo ke znatelnému pokroku (Česká televize, 2017).

Úloha programátorů v případě umělé inteligence by pak nespočívala v programování samotném, ale v usměrňování umělé inteligence a jí vytvořených programů tak, aby si nevytvářela korelace, které jsou mylné či falešné. Naposledy k takovému *„omylu“* došlo, když Microsoft vypustil na síť AI chatbota[7]. Microsoft tak učinil proto, aby lidé celého světa mohli novou umělou inteligenci učit novým



Obrázek 7: Google AI Projects. Zdroj: Google.com

[6] AI – Artificial Intelligence, anglický výraz pro umělou inteligenci.
[7] Chatbot – spojení slov *„chat“* = krátká komunikace/rozhovor a bot = robot. Chatbot je tedy robot/automat určený ke komunikaci s lidmi skrze krátké online zprávy.

poznatkům. Situace se ale vymknula kontrole a z AI chatbota se po chvilce stal agresivní rasista, kdy samovolně postoval[8] na síť nenávistné vzkazy (Business Insider, 2016).

Náhledem do budoucnosti bychom ukončili kapitolu *„historie ve světě programování“* a přesunuli bychom se ke způsobům a postupům, kterými se samotné programování a vytváření softwaru řídí. Tuto problematiku řeší tzv. paradigmata programování.

[8] Post – anglický výraz označující odeslání. v online světě pak odeslání/zveřejnění jisté informace.

2. Metodika výuky programování

Kapitola metodiky výuky programování popisuje jednotlivé přístupy ve výuce programování, tzv. programovací paradigma. Každý způsob a přístup k programování má svoje pozitiva i negativa, podle nich se pak může budoucí programátor rozhodnout, který způsob je mu nejbližší. I když by měl programátor cvičit a vylepšovat hlavně jeden způsob, neměly by mu být zároveň cizí ani ostatní způsoby. Dobrý programátor by se měl umět vyznat i v jiných paradigmatech a v případě nutnosti *„přesedlat“* na jiný programovací jazyk/způsob programování. Odvětví informatiky se rozvíjí velice rychlým tempem, očekávaná je velká flexibilita s variabilitou a tomu by se měl umět programátor přizpůsobit.

Při výuce programování je velice nutné dodržovat obecné zásady, postupy a cíle, tak jako v každé jiné výuce. Výuka většinou probíhá samostudiem, programování je třeba stále opakovat, rozvíjet a vylepšovat. Při výuce programování na školách však může nastat problém, kdy je učitel znalý programátor, ale nemusí vůbec rozumět didaktice. Takový učitel pak vyučuje programování způsobem, jakým byl on sám vyučován, nebo takovým způsobem, jakým programování sám pochopil, takový styl výuky ovšem nemusí vyhovovat každému. Žáky, které učí takový učitel, pak programování nemusí bavit, nebo hůř, nemusí ho pochopit zcela vůbec. Mimo to je dobré, aby programátor učitel nejen vyučoval učivo, ale také sledoval nejnovější trendy, kterými se programování postupně v čase ubírá (ACM, 2001).

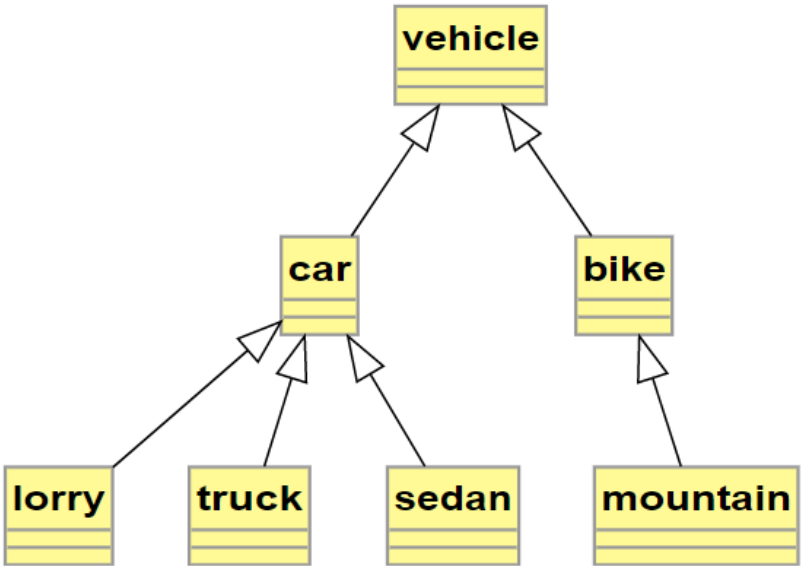
2.1. IMPERATIVE-FIRST, PROCEDURÁLNÍ PROGRAMOVÁNÍ

Jeden z nejsnadnějších způsobů, jakým lze programovat je Imperative-first, kdy se programování podobá funkci procesoru, který prochází kód a plní jednotlivé funkce jednu po druhé, tak jak jsou v kódu zapsané. Stejně tak potom programátor píše krok po kroku, co se má udělat. Imperative-first vyžaduje znalost syntaxe daného jazyka, nicméně pokud se s ní programátor seznámí, je možné bez obtíží psát kód. Je to totiž jedna z nejjednodušších možností, jak programovat. Nevýhodou je dlouhý, někdy možná až nepřehledný kód a také fakt, že pokud se (začínající) programátor naprosto sžije s Imperative-first, může mu pak dělat problém programovat v objektech, kdy program není psán lineárně v jednom programu, ale je zapotřebí více *„částí“* jednoho fungujícího celku. Výhodou je naopak to, že se kód a jeho algoritmus dá dobře přepisovat a je jednodušší jeho optimalizace, tzn., že není potřeba hledat chyby. Celý program je v jednom kódu a při optimalizaci člověk nemusí přemýšlet, zda není jedna část závislá na jiné, protože kód neobsahuje jiné cizí části nebo objekty programu. Imperative-first se kvůli své jednoduchosti používá hlavně na středních školách, kdy je možné již po krátké době a rychlém seznámení s programovacím jazykem vidět výsledky práce (ACM, 2001).

2.2. OBJECTS-FIRST

Objects-first je způsob výuky programování, kde je kladen důraz na principy objektově orientovaného programování. Programátor se tedy nejprve učí sestavovat návrh objektově orientovaného programu, kdy si vytvoří schéma, jak program bude následně vypadat a podle výsledku se pak programuje (ACM, 2001).

Výuka objektově orientovaného programování by měla určitě zahrnovat také seznámení s pojmy zapouzdření, dědičnost a polymorfismus, což jsou 3 základní pilíře, na kterém objektově orientované programování stojí (Úvod do objektově orientovaného programování v C#, 2012). Výhodou i nevýhodou objektově orientovaného programování je fakt, že pokud programátor zná jiný způsob programování (např. Imperative-first), muže mu OOP přijít složité. Pokud se začínající programátor seznámí se způsobem, jak OOP funguje a až potom se učí jeho syntaxi a zásady, nemusí dojít k nepochopení způsobu tohoto programování (ACM, 2001).



Obrázek 8: Příklad OOP a závislost jednotlivých objektů. Zdroj: voho.eu

2.2.1. Aspektově orientované programování

Aspektově orientované programování je rozšíření objektově orientovaného programování o tzv. aspekty, které mají za cíl zvýšit modularitu programu. Program je rozdělen na části, které spolu spolupracují, ale navzájem se kříží co nejméně. Příkladem může být vložení přídavného kódu, který mění chování jiné části programu bez nutnosti změnit již původní existující kód (ACM, 2001).

2.3. DECLARATIVE-FIRST

Metoda programování Declarative-first se zaměřuje na úkol programu, tedy co má být vyřešeno, namísto toho, jak se to má řešit. Deklarativní způsob programování je opakem imperativního způsobu. V tomto případě se specifikuje cíl – zjednodušeně lze říci, že se programuje

tak, aby počítač věděl, co se má vyřešit, namísto toho, jak se to má vyřešit. Celá algoritmizace je tak ponechána interpretovi jistého jazyka (Vermeluen, 2013).

2.4. FUNCTIONAL-FIRST

Functional-first je podmnožina deklarativního způsobu programování uvedeného výše. Metoda programování Functional-first se zaměřuje opět na úkol programu, tedy znovu na to, co má být vyřešeno namísto toho, jak se to má řešit. Tato metoda vytváří výstup na základě vyhodnocení matematických funkcí (ACM, 2001).

Metoda funkčního programování je způsob stavby počítačového kódu, který se v konečném důsledku chová jako při vyhodnocení matematické funkce. Program je postaven na definicích nikoliv na příkazech – opět tedy to, co se má řešit, a ne jakým způsobem. Funkcionální programování je hojně používáno ve výpočetních strojích. Je jednoduché, co se vizáže kódu týče (kód připomíná matematickou rovnici), to ale může způsobit problém pochopit kód, protože obsahuje nízký podíl abstrakce. Programování functional-first dále rozděluje programovací jazyky na *„čisté“* a *„nečisté“*, kdy některé programovací jazyky obsahují v syntaxi čistě definice pro výstup (Adga, Clear, Mercury) a programovací jazyky *„nečisté“*, které dokáží pochopit definici kódu programu (tváří se jako functional-first), ale pracují na bázi Imperative-first, např. jazyky C++, C#, Java a další (ACM, 2001).

2.4.1. Functional-first – programovací jazyky z vývojářského pekla

Výhodou programu functional-first je možnost krátce zapsat kód programu pro řešení zadaného problému. Z jistého, pro někoho možná nelogického, důvodu začaly vznikat i experimentální programovací jazyky, které měly mít za cíl co nejkratší možný kód, a také i co nejmenší možnou velikost datového souboru. Tyto programovací jazyky však *„velkou díru“* do světa neudělaly a mnozí, i zkušení, programátoři jej nazývali *„programovacím peklem“*. Po jejich vyzkoušení se programátoři rádi vrátili k původním standardním rozšířeným (Tišnovský, 2016).

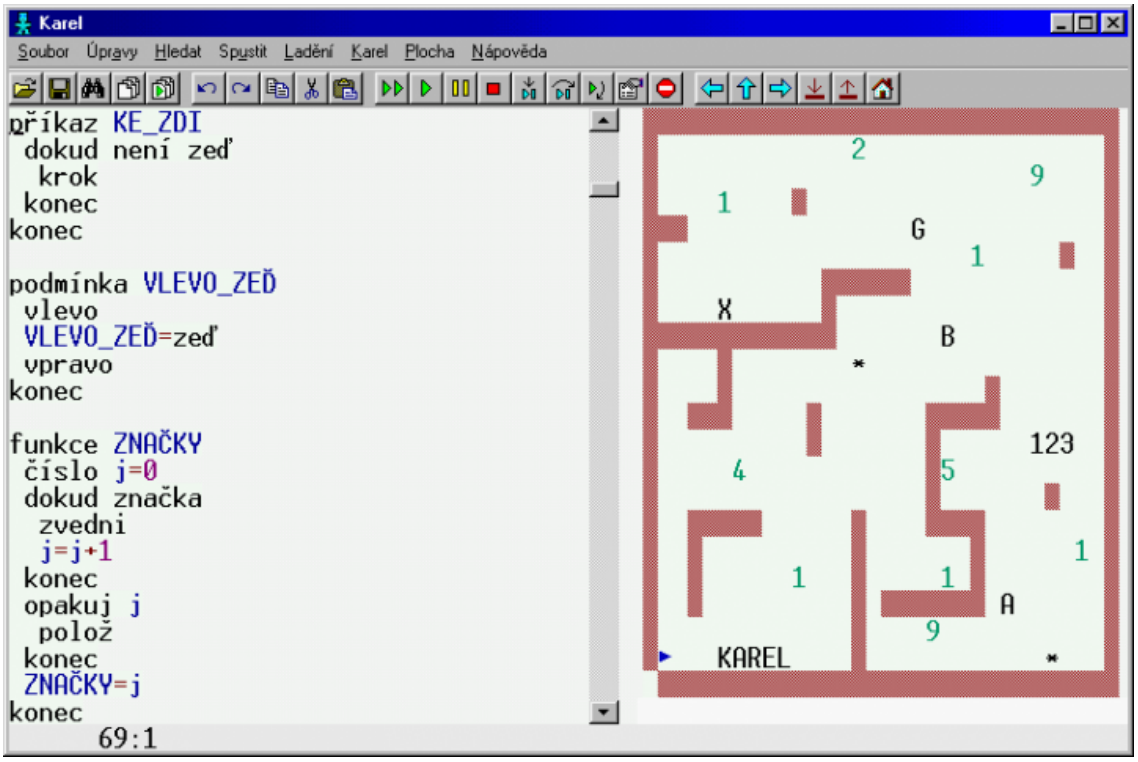
```
99 9[1-$][\$@$@$@$@\/*=[1-$$[%\1-$@]?0=[\$.' ,\]?]?]#
```

Obrázek 9: Programovací jazyk FALSE a zápis algoritmu pro výpočet prvočísel ležících v rozsahu od 0 do 100. Zdroj: root.cz

2.5. ALGORITHMS-FIRST

Způsob výuky programování pomocí Algorithms-first je nejjednodušším způsobem, jak se programování naučit. Metoda spočívá v tom, že je výuka realizována v programovacím pseudo-jazyce. Programovací pseudo-jazyk neslouží k běžnému vytváření kódů ani programů, obsahuje zjednodušené prostředí, příkazy a syntaxi, kterou je však nutno dodržet. Začínající programátor tak pochopí nutnost některých postupů, zároveň však nemusí znát velké množství celkových možností tohoto jazyka – jde pouze o schopnost naučit se obecné zásady programování. Naučené zásady pak lze aplikovat při užívání opravdových programovacích jazyků (ACM, 2001).

Typickým příkladem Algorithms-first programovacím jazykem může být programovací jazyk KAREL[9]. KAREL je robot, který se pohybuje po čtvercové šachovnici, zná pár základních příkazů (otoč se, jdi vpřed, polož bzučák, zvedni bzučák, vypni se) pomocí nich začínající programátor robota ovládá a učí se tak syntaxi, přičemž si je vědom nutnosti ji dodržet, v opačném případě totiž program zahlásí chybu. KAREL se používá na Stanfordské univerzitě, kdy se studenti po pár týdnech s KARLEM postupně přesouvají k programovacímu jazyku Java (Pattis, 1995).



Obrázek 10: Prostředí programovacího jazyka Karel. Zdroj: root.cz

Programátor si tedy vybral svůj hlavní *„styl“* programování a má určité povědomí i o ostatních možnostech, kterými lze software vytvářet. Pokud už software tvoří, vytváří tím hodnotu a za svoji práci by měl být patřičně odměněn. Kapitolu o paradigmatech tedy opustíme a navážeme dále kapitolou o trhu práce v oblasti IT a programování.

[9] Programovací jazyk je nazván po Karlu Čapkovi, jakožto odkaz na člověka, který toto slovo prvně použil

3. Trh práce a programování

3.1. VÝVOJ TECHNOLOGIE VS. „VÝVOJ PROGRAMÁTORŮ“

Moorův zákon hovoří o tom, že vývoj technologií roste exponenciálně. Původní přesné znění zní: *„počet tranzistorů, které mohou být umístěny na integrovaný obvod, se při zachování stejné ceny zhruba každých 18 měsíců zdvojnásobí.“*. Zhruba každé dva roky se tedy zdvojnásobí počet tranzistorů na integrovaném obvodu a tím se zvýší i výpočetní výkon[10] (Moore, 1965).

Problémem zde není vyrobit co dva roky výkonnější a silnější integrované obvody, ale najít k nim lidi, kteří by pak mohli výsledné produkty programovat. K otázce: *„Proč se více lidí neživí jako programátoři?“* z webu Quora.com odpověděl Brian Feldman, konzultant pro vývoj a robotický specialista, následovně: *„Stát se dobrým programátorem je neskutečně složité a trvá to nesmírně dlouho.“* Svoji odpověď pak rozvinul ještě tvrzením, že nemůžeme zasadit pár stromů a očekávat 2000letý les, který by vyrostl přes noc. Dále hovořil o tom, že programátoři většinou nevycházejí samostatně ze škol, ale že se musí učit i sami (jakákoliv škola samotné programátory nevytváří), dlouho trénovat a z krátkých programů pak plynule přecházet k dlouhým skriptům, kdy vypilování této činnosti trvá roky. Jako programátor nejen vytvářet, ale i opravovat nedokonale vytvořené programy a vytrvat v tom i při opakovaných neúspěších. Mít znalosti z matematiky a logiky a umět je aplikovat. Mít skvělou krátkodobou i dlouhodobou paměť. Znát hardware, pro který programuji. Znát software, pro který programuji. V neposlední řadě také mít *„ohromný blok čistého volného času“* (interpretováno jako *„neomezený“* čas, kdy se programátor prostě nezvedne od počítače, dokud nevzejde pozitivní výsledek). Po dlouhé, vyčerpávající odpovědi pak konstatoval, že ne každý má na to, aby se stal výborným programátorem. Je mnoho *„lacinýchˡ¹¹“* programátorů, ale jenom hrstka opravdu **dobrých** programátorů (Feldman, 2014).

Co také souvisí s vývojem v čase, je skutečnost, že dříve si programátor vystačil s pouhou znalostí HTML, uměl třeba i ve FORTRANu, či BASICu a přidal k tomu jako bonus CSS, o práci pak bylo postaráno. V dnešní době je ale situace trošku komplikovanější a zákeřnější. IT sektor i firmy obecně mají daleko vyšší požadavky, kdy samotné řádky jednoduchého kódu nestačí a v kurzu je OOP a práce s velkým objemem dat. Dnešní svět je totiž zaměřen na práci s velkým objemem informací, které jsou spolu provázány, a které na sebe navazují. Samotný program nebo aplikace na webu pak nemá pouze za cíl fungovat, ale i získávat data, ty dále zpracovávat a na základě zpracování interpretovat (Navrátil, 2003).

[10] Součástky lze buď zmenšovat, při zachování stejného výkonu nebo zachovat aktuální velikost a zvýšit výkon. Souvisí to s miniaturizací celého celku nebo jenom miniaturizací určité části elektronické součástky (pozn. autora)

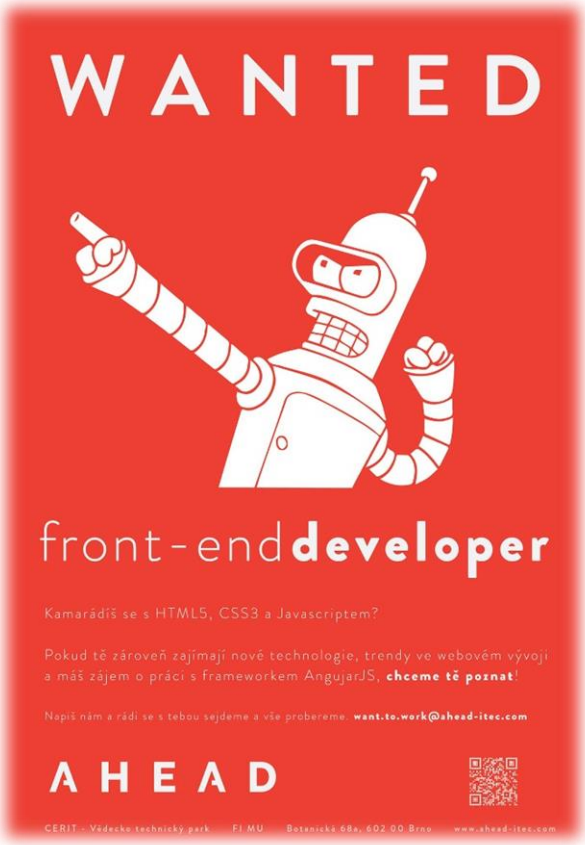
[11] z kontextu úplné nezkrácené odpovědi vyplynulo, že autor naznačil, že mnoho lidí se za programátory považuje už jen tím, že napíší program *„Hello World!“* – program, který po spuštění pouze vypíše na obrazovku právě zmíněný nápis *„Ahoj světe!“*

3.2. HON NA PROGRAMÁTORY NA TRHU

Na webu i kdekoliv jinde je možné v dnešní době vidět, že se firmy snaží aktivně hledat programátory. Z předchozího odstavce bylo možné vyčíst, že *„programátorů“* je dost, ale kvalitních a dobrých programátorů je opravdu málo. Programátor ale není člověk univerzální, a ne vždy platí pravidlo, že programátor, kterého si firma našla, splňuje kritéria, která jsou důležitá pro výkon jeho pracovní činnosti. V tomto případě jde hlavně o zaměření programátora, kdy se ve většině případech jistý programátor věnuje pouze jednomu programovacímu jazyku (může např. umět i jiné, nebo jim rozumí, ale není v nich tak zběhlý). Firma, která hledá programátora Javy, pak dobrého programátora jazyka C nejspíše odmítne, protože potřebuje právě zmíněnou Javu. Existují výjimky, kdy např. jeden člověk perfektně zvládá více programovacích jazyků, takových programátorů je však velice málo (Kladivová, 2016).

Firmy ale většinou požadují po kandidátovi mimo znalosti programování také znalost anglického, nebo německého jazyka. Co však problémem není, je fakt, že firmám nejde primárně o diplomované jedince – stačí jim, když jistý člověk opravdu programovat umí, kdy součástí přijímacího pohovoru není jenom interview, ale i nějaký problém, který by měl kandidát o práci zvládnout na místě vyřešit, např. naprogramovat jednoduchý program, který vyřeší jednoduchý úkon zadaný firmou (IT Profese, nedat.).

Problém, který je spjatý s nedostatkem programátorů ve firmách, také souvisí s tím, že některé firmy leckdy nejsou ochotné dobrého programátora opravdu dobře zaplatit, jindy např. nemusí ani vadit finance, jako třeba pracovní kolektiv, motivace k práci nebo jiné nefinanční benefity (Dresler, 2014).



Obrázek 11: Náborový plakát programátorů firmy AHEAD iTec . Zdroj: AHEAD iTec

3.3. ZAČÍNAJÍCÍ PROGRAMÁTOR A ŠANCE ZISKU PRÁCE

Dnešní trh práce není zaměřen pouze na jeden programovací jazyk. Velká heterogenita dává možnost v podstatě každému programovacímu jazyku, avšak některé jsou žádanější než jiné. Velkou roli zde hraje platforma, pro kterou je programovací jazyk vhodný. Některé programovací jazyky

jsou vhodné pouze pro určitá prostředí, jiné programovací jazyky zase sází na multiplatformní podporu, kdy v podstatě užití není takřka ničím omezeno. Níže uvedená tabulka znázorňuje programovací jazyky a jejich zastoupení na pracovním trhu za rok 2017/18 (GitHub, 2017)

Obrázek 12: Zastoupení programovacích jazyků na trhu ve světě. Zdroj: GitHub.com

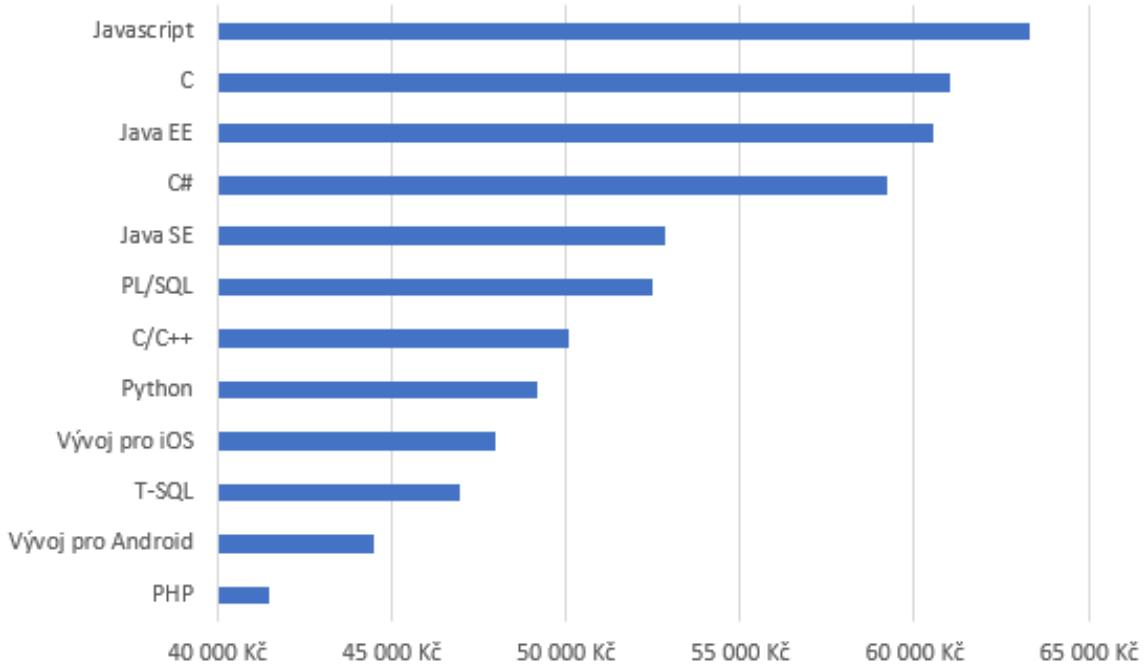
Z tabulky vyplývá, že nejoblíbenější a nejvíce zastoupený je programovací jazyk JavaScript. JavaScript (pozor, není to samé, co Java!) je programovací jazyk, který vznikl hlavně pro potřeby programování webových aplikací a je možné ho přímo vkládat do HTML kódu (Janovský, nedat), jelikož se dnes prakticky *„všechno“* nachází na internetu, je jasné, že JavaScript si drží právem první místo.

Po JavaScriptu následovala v minulosti Java, tu ale nedávno předběhl Python. Je tomu tak proto, že Python je od jisté doby standardně přítomen v Linuxu i iOS[12] (včetně MacOS, v prostředí Windows se ještě standardně nenachází, avšak jeho instalace není vůbec obtížná), přičemž Javu je nutné do systému vždy prvotně nainstalovat a poté ještě někdy ručně nastavovat – je proto logické, že je pohodlnější použit programovací jazyk, který má pro svoji funkci již od začátku vše připraveno a ke svému běhu nic jiného nevyžaduje. Python navíc rozumí a podporuje různá programovací paradigmata (včetně OOP, imperativního, deklarativního či funkcionálního) a je zdarma. Java (také zdarma, také multiparadigmatická) se tedy musela smířit se třetím místem, ale i přes to je to stále oblíbený programovací jazyk, který toho může mnoho nabídnout (Comparing Python to Other Languages, nedat.).

Ruku v ruce jde s programátorskou profesí i odměna za vykonanou práci. Jak již bylo řečeno, dobrých programátorů je málo a firmy, pokud o programátora nechtějí přijít, si je snaží udržet nejen

[12] Operační systémy firmy Apple, MacOS pak přímo operační systém pro stolní počítače Apple

dobrým platem (Hospodářské Noviny, 2015). Níže uvedená tabulka znázorňuje platy programátoru v ČR v roce 2017.



Obrázek 13: Platy programátorů v ČR v roce 2017. Zdroj: ittalents.cz

Na první příčce se nachází JavaScript s průměrnou mzdou programátora kolem 63 000,-, na druhém místě se zde ale neumístil Python, nýbrž programovací jazyk C. I přes značnou oblibu (ve světě) jazyka Python není ohodnocen tak dobře, jako méně oblíbené jazyky C a jeho následné bratry C++ a C# po kterých je v České republice větší sháňka, a proto jsou firmy ochotny zaplatit za programátory těchto jazyků více. Úplně nejhůř se pak umístil PHP, nicméně žádný z dotázaných programátorů nebral podle zaznamenaných údajů méně, jak 40 000,- za měsíc (ITtalents, 2017).

Z výše uvedeného lze vyčíst, že programátoři jsou na tom po finanční stránce daleko líp, než je průměrná mzda v České republice, která činí podle posledních statistických průzkumů něco málo přes 31 000,- (Průměrné mzdy 3. čtvrtletí 2017, 2017). V tomto případě je ale důležité podotknout, že většina obyvatel na průměrnou mzdu vůbec nemusí dosáhnout a medián by měl v tomto případě daleko přínosnější informační hodnotu a ten činil 25 181,-.

I přesto, že mají programátoři daleko větší finanční ohodnocení než většina obyvatel žijící v ČR, moc lidí se programováním neživí. Důvody pro to byly vyjmenovány výše. Jak to tedy udělat, aby měli lidé v budoucnu větší šanci se programátorem stát? Pokud se lidé s programováním začnou seznamovat dříve a stane se součástí výuky na školách, mají budoucí absolventi větší šanci programování lépe pochopit, uchopit jeho podstatu, rozvíjet se v této oblasti, a neztrácet tak drahocenný čas. Na tuto problematiku navazuje další kapitola, která řeší samotný přístup k ICT a programování na školách v ČR.

4. Výuka programování na školách

4.1. ZÁKLADNÍ ŠKOLY A RVP

Rámcový vzdělávací program pro základní vzdělávání je rozdělen na 1. a 2. stupeň. Oblast programování bychom mohli najít ve vzdělávací oblasti *„informační a komunikační technologie“*. Vzdělávací oblast *„informační a komunikační technologie“* se nachází jak v 1. tak ve 2. stupni RVP ZV, nicméně je pouze upřesněno, co obsahuje 1. stupeň a co stupeň 2. Nikde není přímo závazně uvedeno, kolik by měla činit časová dotace a v jakém ročníku by se mělo s předmětem začít (RVP ZV, 2017).

První stupeň RVP ZV pouze vyžaduje učivo obsahující:

- základní pojmy informační činnosti – informace, informační zdroje, informační instituce
- struktura, funkce a popis počítače a přídavných zařízení
- operační systémy a jejich základní funkce
- seznámení s příponami souborů (doc, gif, jpeg, wav…)
- multimediální využití počítače
- jednoduchá údržba počítače, postupy při běžných problémech s hardwarem a softwarem
- zásady bezpečnosti práce a prevence zdravotních rizik spojených s dlouhodobým využíváním výpočetní techniky

Z výše uvedeného vyplývá, že se žáci na prvním stupni pouze s fungováním počítače seznamují, žádné programování, byť ani v jednoduché formě, v RVP ZV předepsáno není. Záleží pouze na školách, zda si do vlastního ŠVP programování vloží, s tím, že stihnou vše předepsané + pro začátek nějakou jednoduchou formu programování, nebo zvolí možnost výuky programování formou volnočasové aktivity či nějakého kroužku.

Druhý stupeň pak obsahuje učivo:

- vývojové trendy informačních technologií
- hodnota a relevance informací a informačních zdrojů, metody a nástroje jejich ověřování
- internet
- počítačová grafika, rastrové a vektorové programy
- tabulkový editor, vytváření tabulek, porovnávání dat, jednoduché vzorce
- prezentace informací (webové stránky, prezentační programy, multimédia)
- ochrana práv k duševnímu vlastnictví, copyright, informační etika

Žáci by již měli umět počítač samostatně obsluhovat a k tomu přidávají postupně učivo další. Uvedené učivo 2. stupně opět explicitně programování přímo nevyžaduje, avšak obsahuje učivo *„internet“* a *„prezentace informací“* a v mnoha případech si školy do ŠVP vkládají do učebních osnov, většinou pro 9. ročníky, učivo týkající se vytváření jednoduchých webových stránek, přičemž

se k této problematice využívá jazyka HTML, popř. HTML5. První programování na základních školách tedy oficiálně můžeme nalézt až na druhém stupni a ve většině případech až ke konci žákovy povinné školní docházky (RVP ZV, 2017).

4.2. Střední školy a RVP

Po absolvování základního vzdělání se ve většině případech očekává, že se žák posune dál na střední školu. V tomto případě ale nastávají problémy spjaté s programováním, protože ne všechny střední školy se zaměřují technickým směrem, nebo přímo na informatiku.

Obecně má střední škola závazné předpisy, které musí podle rámcového vzdělávacího programu pro střední školy plnit. V tomto případě ale existuje více jak 100 rámcových vzdělávacích programů pro střední školy na základě zaměření studia.

Střední školy s humanitním zaměřením pak programování, jakožto součást *„informační a komunikační technologie“*, ve svém školním vzdělávacím programu *„ořezávají“* na minimum ve prospěch jiných předmětů, které si škola zvolí (cizí jazyky, výtvarná umění, hudební zaměření, praktické pracovní zaměření).

Gymnázia jsou v tomto ohledu svým způsobem specifická. Byť jsou specializované buď humanitním, nebo technickým směrem, mají i přes to učivo rovnoměrněji rozložené (u absolventa gymnázia se počítá s tím, že bude pokračovat studiem na vysoké škole, přičemž si tito lidé většinou chtějí nechat cestu otevřenou jak humanitním směrem, tak druhým, technickým směrem) a vzdělávací oblast *„informační a komunikační technologie“* se vyučuje s větším *„nadšením“*, než na středních školách zaměřených humanitním směrem.

Střední školy technické nebo přímo zaměřené na informatiku jsou potom pravým opakem středních škol s humanitním zaměřením. Na těchto školách se obecně oblast *„informačních a komunikačních technologiích“* nepodceňuje. Tyto školy se pak buď zaměřují na hlubší a kvalitnější výuku programování v jednom programovacím jazyce, nebo vyučují základy programování v různých (někdy i odlišných) programovacích jazycích.

Způsob, jakým střední škola provádí výuku programování, je pak součástí výzkumné praktické části této diplomové práce.

Kromě středních škol probíhá výuka programování i na školách vysokých. Tato kategorie však nespadá pod oblast regionálního školství a cílem kvalifikační práce není analýza tohoto jevu, proto bude tato kategorie vynechána.

4.3. Jiné formy vzdělání spjaté s programováním

Programování se v České republice objevuje nejdříve v 9. ročníků základních škol a někdy až na středních školách, což je podle některých expertů pozdě. Programovat by se měli učit už děti,

které zvládnout číst a obsluhovat chytrou elektroniku (tablet, chytrý telefon[13], notebook). Programování, které by se však začalo vyučovat již na prvním stupni základních škol je spíše utopií. Proto vznikají volnočasové aktivity a kroužky pro děti, které nemají za cíl udělat z mladého člověka do 9. ročníku prvotřídního programátora, ale naučit ho myslet tak, aby pro něj jednou programování nebylo obtížné. Kroužky a volnočasové aktivity v tomto případě používají pseudo-programovací jazyky (viz algorithms-first paradigma), kdy se dítě učí základy programování formou hry (Bajtler, 2017).

Obdobná situace pak nastává i u starších lidi, kteří již mají školní docházku dávno za sebou, ale nechtějí si nechat *„vlak informatiky"* ujet. Takoví lidé se mohou učit programovat také skrze různé kurzy, nebo se mohou naučit programovat doma. Postačí jim k tomu připojení k internetu, portál, který se věnuje výuce programování, a hlavně chuť naučit se něco nového (někdy bývá zapotřebí také znalost anglického jazyka, protože pro programování je specifický právě anglický jazyk – respektive jeho jednoduchá forma při dokumentaci programů). Příkladem pak mohou být portály W3schools.com nebo různé projekty organizace Czechitas[14].

4.4. PROGRAMOVÁNÍ VS. INFORMAČNÍ GRAMOTNOST

Informační gramotnost je znalost a schopnost člověka identifikovat informační potřeby, zvolení nejoptimálnějšího řešení zisku informací, užití odpovídajících zdrojů a informačních systémů, schopnost filtrovat pouze požadované a relevantní informace, získané informace kriticky zhodnotit, informace dále vhodně zpracovat a využít, dokázat informace zprostředkovat jiným lidem i v jiných podobách prostřednictvím různých technologií a posoudit morální a právní aspekty využívání informací (Dostál, 2007).

Z výše uvedeného vyplývá, že člověk informačně gramotný si umí potřebné informace vyhledat a relevantně s nimi zacházet. S informační gramotností pak dále souvisí počítačová gramotnost a gramotnost digitální – člověk užívá počítač a digitální technologie k zisku a využití informací. Zde slovo **užívá,** nabírá zásadního významu, protože programování samotné není pouze užívání, ale již **vytváření** něčeho zcela nového. Člověk počítačově gramotný bude způsobilý k užívání již vytvořeného programu, ale nevypovídá to nic o tom, jestli by uměl podobný program sám naprogramovat, nebo jestli by uměl naprogramovat jenom jeho část (či vůbec něco naprogramovat). Je tedy jasné, že samotné programování nemusí mít s výše uvedenými gramotnostmi nic společného (pozn. Naopak to ve většině případů platí – programátor je člověk počítačově, informačně i digitálně gramotný). Tuto problematiku se snaží řešit MŠMT chystanou změnou, která se zve strategie digitálního rozvoje do roku 2020, která má za cíl na školách vychovávat nejen lidi informačně gramotné, ale i lidi, kteří by po absolvování základního a středního vzdělávacího cyklu měli povědomí i o samotném programováním.

[13] Chytrý telefon se od *„hloupého"* liší tím, že obsahuje otevřený operační systém (Android, Windows Phone, iOS). Dříve byly mobilní telefony od výrobce *„uzavřeny",* tj. nedaly se do nich instalovat další aplikace.
[14] Česká nezisková organizace s cílem vzdělávat dívky a děti v oblasti IT.

4.5. STRATEGIE DIGITÁLNÍHO VZDĚLÁVÁNÍ DO ROKU 2020

Strategie digitálního vzdělávání do roku 2020 je návrh vydaný MŠMT. Obsahuje řadu nových způsobů a strategií, jak postupně proměnit léty zakořeněnou metodu výuky na základních a středních školách. Nové metody a způsoby mají reagovat na změnu a vývoj digitálních technologií. Pokud se celý svět ubírá jistým směrem, mělo by vzdělávání v ČR jít směrem stejným – v tomto případě se jedná o zapojení moderních technologií přímo do výuky (STRATEGIE DIGITÁLNÍHO VZDĚLÁVÁNÍ DO ROKU 2020, 2014).

Součástí strategie je také sledování aktuálního stavu vybavenosti škol, co se digitálních technologií týče. Školy většinou digitálními prostředky do jisté míry disponují (většinou mají vybavení z dob minulých projektů, kdy se školy začaly vybavovat technickými prostředky), ale tyto prostředky jsou leckdy i víc jak 5 let staré, což chce nová strategie také změnit pravidelným obměňováním technických a digitálních prostředků (STRATEGIE DIGITÁLNÍHO VZDĚLÁVÁNÍ DO ROKU 2020, 2014).

Součástí tohoto dlouhodobého plánu je také zařazení informačních a komunikačních technologií i jako průřezového tématu. Žáci se s ICT[15] setkávají už na prvním stupni, ale i přes všechnu snahu zůstává předmět *„informatiky“* na školách většinou v pozadí, kdy se v rámci výuky stále opakuje uživatelské užívání kancelářských balíků Microsoft Office a hardwarové vybavení počítače. Při průměrné dotaci 1-2 hodin *„informatiky“* týdně pak na nic jiného důležitějšího nezbyde čas. Průřezové téma ICT pak může fungovat tak, že by se např. v hodinách českého jazyka mohl žák naučit používat textový editor, v hodinách matematiky/fyziky pak tabulkové procesory, tvorba map a GIS by mohla být součástí zeměpisu/geografie, práci s grafickými editory by mohly částečně převzít hodiny výtvarné výchovy, střih zvuku pak hudební výchova (tzv. estetické výchovy by si pak mohly vzít na práci tvorbu videoklipů, včetně návrhu, produkce, marketingu apod.). ICT by se tak mohlo zapojit do kteréhokoliv předmětu, samotná *„informatika“* by pak tento celý komplex mohla doplňovat, a navíc se věnovat samotnému programování, které chce MŠMT právě zmíněnou strategií do oblasti ICT doplnit. V tomto případě totiž není cílem vychovat *„mračna“* nových programátorů, ale pomocí programování naučit děti přemýšlet v jiné rovině. (Paulenková, 2017).

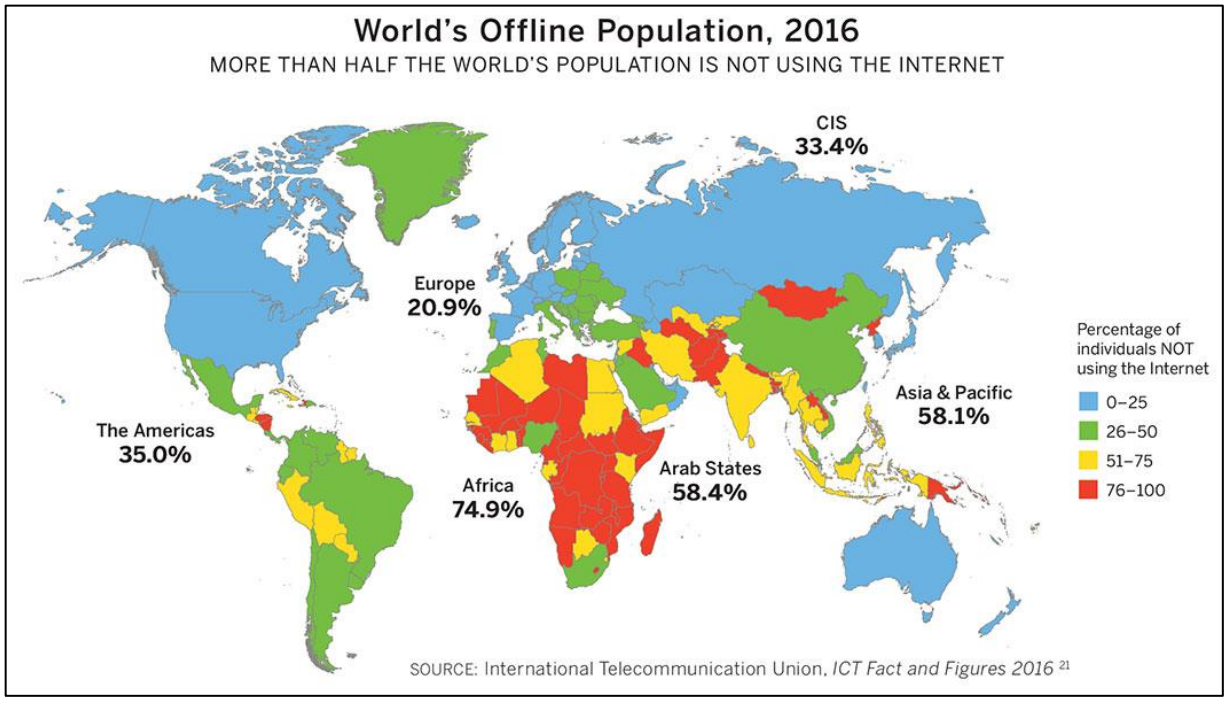
Mimo samotné vzdělávání žáka ale musí také dojít ke vzdělávání učitelů. Rekvalifikace by se týkala v podstatě všech učitelů, kteří (pokud by neuměli) by se naučili techniku obsluhovat, aby mohli skrze digitální technologie nejen vyučovat, ale také řídit výuku, aby následně i samotní žáci mohli aktivně spolupracovat a s pomocí digitálních technologických prostředků sami tvořit. Největší a nejtěžší část by pak tvořila rekvalifikace učitelů informatiků, kteří by měli zprostředkovávat žákům výuku programování. Obecně platí, že na školách se dnes pohybuje jenom velice málo učitelů, kteří by uměli dobře programovat, a to hlavně z toho důvodu, že pokud už programovat dobře umí, většinou odejdou do nějaké firmy, která jim za jejich práci zaplatí několikanásobně více, než škola (STRATEGIE DIGITÁLNÍHO VZDĚLÁVÁNÍ DO ROKU 2020, 2014).

[15] ICT – zkratka pro informační a komunikační technologie.

MŠMT již strategii naplánovalo, nyní je potřeba ji dobře zorganizovat a správně řídit. V konečném důsledku by žák, který by absolvoval vzdělání touto cestou, byl připraven na život v digitální době 21. století, kdy: *„je nejen potřeba digitální technologie obsluhovat, ale i tvořit a programovat.“* (citát, Stephen Hawking, nedat.)

4.6. DIGITÁLNÍ PROPAST

Společně s programováním a jeho možnostmi přichází však také velká překážka. Digitální propast je pomyslná propast znázorňující rozdělení lidí na dvě skupiny. Jednu skupinu tvoří jedinci, kteří disponují přístupem k technologiím (jako je třeba i internet) a druhou skupinu, která tento přístup nemá. Digitální propast je způsobena ekonomickou a sociální nerovností, s tím také souvisí právě užití a znalost informačních a komunikačních technologií. Programování, jakožto součást informačních a komunikačních technologií s tím pak přímo souvisí – lidé, kteří nedisponují těmito technologiemi, určitě nebudou v této činnosti zběhlí. Když se k tomu připočte aktuální potřeba programátorů a programování, tak v konečném důsledku těmto lidem *„ujíždí“* technologický vlak, a i možnost finančního výdělku, který by v těchto oblastech mohl zvýšit životní úroveň. Platí zhruba rozdělení na *„online připojený sever“* a *„off-line jih“* (je to podobné jako environmentální stratifikace jako bohatý sever VS. chudý jih) (Norris, 2001).



Obrázek 14: Mapa off-line světa v roce 2016. Zdroj: ITU.int

Po teoretickém výčtu faktů týkajících se samotného programování, které jsou obsaženy v dokumentech RVP, bychom se přesunuli k praktickému výzkumu zkoumající situaci programování na středních školách. Bude tak možné vypozorovat, jakým způsobem doopravdy výuka programování probíhá a jaká je její skutečná současná úroveň.

5. Výzkum – výuka programování na středních školách ČR

RVP dává školám určité hranice, které škola musí dodržovat, avšak školy mají v tomto případě i jistou volnost a mohou si vytvořit své vlastní plány na průběh výuky. Takový plán se zve ŠVP a škola se právě tímto dokumentem, který popisuje, jak výuka na jisté škole funguje, může profilovat a od jiných škol lišit a zvyšovat konkurenceschopnost tím, že nabízí něco jiného, co je např. žádanější. Ve výzkumné části právě proto zjišťujeme, jaká je aktuální skutečná situace výuky programování na školách. Cílem výzkumu je srovnat činnost programování *„včerejška, dneška a zítřka"* v ČR na středních školách a gymnáziích. Zjistit, jak se programovalo na těchto školách dříve, kdy výuku stanovovaly osnovy, tedy před zavedením RVP a ŠVP a jak je vedena výuka programování v současnosti. Protože je programování velice důležitou součásti dnešní informační doby a význam programování neustále roste, chceme se zaměřit i na to, jak se k tomuto tématu střední školy staví dnes a co chtějí do budoucna ve vzdělávací oblasti informačních a komunikačních technologiích, zvláště v programování, změnit.

5.1. Dílčí cíle výzkumu:

1) Zhodnotit stav výuky programování každé dotazované školy.
2) Porovnat výuku programování jednotlivých škol podle jejich zaměření (obecné, jazykové, sportovní VS. technické).
3) Navrhnout možné změny ve výuce ICT a v přístupu k programování.

5.2. Popis výzkumu

Výzkum spočívá v analýze ŠVP středních škol a gymnázií, přičemž se zajímáme o výuku programování v oblasti výuky ICT. V rámci výzkumu charakterizujeme jednotlivé školy, kdy je škola nejprve stručně představena a v krátkosti je popsáno, jakým směrem se škola ubírá. Dále zkoumáme, jestli vůbec probíhala výuka programování před zavedením RVP a povinného vytváření ŠVP a jakým způsobem byla výuka realizována. Dále zjišťujeme, jaký je stav a podoba výuky programování na školách v aktuální době, se zaměřením na prostředí výuky a časovou dotaci. V rámci strategie digitálního rozvoje vzdělávání do roku 2020 vydaného MŠMT se dále zabýváme tím, jaké kroky škola hodlá podniknout pro to, aby se programování dostalo více do povědomí žáků a ostatních lidí. Zajímá nás, jakým způsobem chtějí školy programování učit, aby dosáhly co nejvyšší efektivity, respektive zda vůbec chtějí provést ve způsobu výuky nějaké změny. Součástí dotazníkového šetření je také zjišťování toho, jestli je možné na škole absolvovat z oblasti ICT a programování maturitní zkoušku. Autor zjištěné informace v práci diskutuje s informacemi uvedenými v teoretické části práce a uvádí zjištěné informace do kontextu s aktuálním stavem v oblasti výuky programování na středních školách a gymnáziích. Autor se dále snaží tento stav kriticky zhodnotit a na tomto základě pak přichází s návrhy na možné úpravy či potřebné intervence v této oblasti do budoucnosti. Výzkum probíhal formou polostrukturovaného interview s učiteli ICT

daných škol, dotazníkem odeslaným skrze email, pokud se škola nenacházela v Olomouckém kraji a také samotnou analýzou ŠVP z oficiálních webových stránek dané školy. Občas ale nastala i situace, že ŠVP nebylo na oficiálních stánkách dané školy k dispozici a nebylo možné nahlédnout ani do fyzické kopie. V tomto případě byly informace dodatečně získány opět z rozhovoru či emailu, v neposlední řadě pak také z webu http://infoabsolvent.cz. Dotazník je k nalezení v přílohách této kvalifikační práce, interview pak tyto otázky obsahovalo také, ale rozhovor poskytoval více času a lepší možnosti reagovat na podněty od dotazovaného a tím podrobněji zmapovat zkoumaný jev.

5.2.1. Formulace výzkumných předpokladů:

Mimo samotné výzkumné šetření pomocí interview, dotazníku a analýzy školních vzdělávacích plánů byly zformulovány i výzkumné předpoklady. Po ukončení výzkumu bude možné tyto výzkumné předpoklady označit jako platné, nebo neplatné. Tyto výzkumné předpoklady navíc lépe vystihnou některé platné skutečnosti vyplývající z aktuálního stavu výuky ICT a programování.

- Aspoň na 70 % všech dotazovaných škol již dříve programování probíhalo.
- Aspoň na 70 % všech dotazovaných škol se mimo výuku ICT také nyní programuje.
- 100 % škol, které jsou ryze technické, vyučovalo programování před zavedením RVP.
- 100 % škol, které jsou ryze technické, dnes vyučuje programování.
- Všechny dotázané školy jsou již seznámeny s plánem Strategie digitálního vzdělávání do roku 2020 a mají zpracovaný plán na změnu v přístupu k výuce programování.
- Poměr výuky programování v rámci ICT se na technických školách a gymnáziích liší.

5.3. ZPŮSOB VYHODNOCOVÁNÍ VÝZKUMU

Vyhodnocování získaných dat probíhalo formou komparace dat za pomocí kontingenční tabulky, kde se mapované jevy vyskytují v čisté podobě: pro čtenáře, kterého pouze zajímá mapovaný jev, pak stačí pouze nahlédnout do této části, kdy je eliminována slovní část *„charakteristiky školy“*, či jiné slovní hodnocení nebo doplňky, a srovnávají se pouze získané výsledky. Pro lepší orientaci jsou tabulky doplněny o komentáře vztahující se k mapovanému jevu a o grafy, které lépe znázorňují celkovou situaci v číslech.

Gymnázium Jakuba Škody, Přerov

CHARAKTERISTIKA ŠKOLY

Gymnázium Jakuba Škody v Přerově je škola, která nabízí studentům jak 8letý cyklus vzdělávání, tak 4letý cyklus. Škola sama sebe prezentuje, jako výběrovou školu, která má na studenty vyšší nároky, skvělé výsledky v různých soutěžích a dokumentace výsledků žáků z České školní inspekce pak toto tvrzení potvrzuje a škola se tak opravdu řadí mezi nejlepší školy v naší republice. Gymnázium není oficiálně vyhraněno pro humanitní studium nebo studium přírodních věd, ŠVP uvádí, že zaměření je všeobecné a směřuje ke kvalitní přípravě žáků na další studium po maturitě. Neoficiálně je pak toto gymnázium známo jako *„matematické“*.

VÝUKA PROGRAMOVÁNÍ DŘÍVE

Programování na této škole probíhalo i dříve, a to v programovacím jazyce ALGOL a PASCAL. Programování nebylo zahrnuto mezi běžné předměty, ale do předmětu *„seminář ICT“*. Tento seminář byl hlavně určen pro studenty, kteří již věděli, že se budou programováním zabývat i v budoucnu. Proto se v semináři neprogramovaly pouze cvičné příklady, ale leckdy i užitečné programy nebo hry. U žáků, kteří si vybrali seminář ICT, se předpokládalo, že budou i z předmětu informatiky maturovat.

ANALÝZA ŠVP, ICT A PROGRAMOVÁNÍ DNES

Škola má v ŠVP v nižším stupni uvedený předmět informatiky – vzdělávací oblast ICT. Žáci se informatice věnují ihned od samotného nastoupení na školu, tj. od Primy (6. ročník základní školy), kdy se s předmětem informatiky setkávají po celou dobu studia nižšího stupně (tj. do Kvarty – 9. ročník ZŠ), ale na nižším stupni se žáci s programováním nesetkávají, probírá se HW a SW[16], kancelářský balík Microsoft Office, internet a jeho využití, multimédia (úprava fotek, sdílení multimédií, vlastní tvorba), bezpečnost při práci s PC a bezpečnost na internetu, OS[17] a jejich rozdělení. Pro vyšší stupeň pak výuka v oblasti ICT obsahuje letmé pokračování a navázání na znalosti z nižšího stupně studia. Programování a algoritmizaci zde najdeme až v Sextě (2. ročník SŠ), kde se programování opravdu jenom *„naťukne“*, ale tím zároveň oficiální výuka ICT končí. Pokud by se chtěl žák dále věnovat informatice a jít více do hloubky, je pro tyto potřeby na škole veden předmět *„Seminář informatiky“*, kde se žák zaobírá tématy spjatými s ICT více do hloubky, a to i včetně programování, aktuálně jsou probírány základy programování ve více programovacích jazycích (HTML, Java, C, PASCAL).

[16] HW = Hardware – hmatatelná složka počítače, SW = Software – nehmatatelná složka počítače.
[17] OS – operační systém, program, který se načítá a zavádí do paměti ihned po startu počítače.

PLÁNY ŠKOLY NA ZMĚNU VÝUKY PROGRAMOVÁNÍ DO BUDOUCNA

Co se týče Strategie digitálního vzdělávání do roku 2020, tak s ním škola počítá. Nicméně velké změny se dít nebudou, škola totiž výuku programování poskytuje. V sextě se vyučují základy programování, dodatečné znalosti lze získat v rámci volitelného předmětu *„seminář informatiky“*. Časová dotace tak zůstane nejspíše nepozměněná. Co však škola hodlá změnit, je přístup k programovacím jazykům. Do nynější doby se výuka programování uskutečňovala/uskutečňuje ve více programovacích jazycích. Probírá se tak jenom základ z každého vyučovaného programovacího jazyka a syntaxe těchto jazyků. Do budoucna se má výuka změnit tak, že bude probírán pouze jeden programovací jazyk, jeho syntaxe a celkové možnosti jazyka do takových mezí, které nepřesáhnou časové možnosti a schopnosti žáků. Učitelé informatiky z gymnázia se svěřili, že nynější plejáda jazyků má být nahrazena jedním, a to nejspíše jazykem C#. Důvod použití jazyka C# je to, že není tak složitý na pochopení a jeho prostředí nepotřebuje serverovou část, jako třeba PHP. I přes různé spekulace se učitelé shodli, že jazyk C# zůstane nejspíše finální volbou.

MATURITNÍ ZKOUŠKA

Studenti mají možnost na škole maturovat z předmětu informatiky z oblasti ICT. U maturitní zkoušky z informatiky se předpokládá, že student si vybral předmět *„seminář z informatiky“* a počítá se proto s tím, že má veškeré potřebné znalosti ke složení zkoušky. Maturitní zkouška se skládá ze 2 částí, a to z části teoretické a praktické. Teoretická část obsahuje 15 témat z oblasti ICT, u této části stačí prokázat znalost teoretickou a téma popsat slovy. Praktická část obsahuje programování na počítači, kdy student musí program vytvořit, včetně komentářů, a slovně popsat, jak program funguje. Na to, aby student maturitní zkoušku z předmětu informatiky udělal, musí úspěšně složit alespoň jednu z částí, tedy teoretickou, nebo praktickou. Pokud má však jenom jednu polovinu, projeví se to na výsledné známce.

Poznámky autora

Volba jazyka C# se zdá možná velice *„neveselou“*. S dnešními možnostmi a velkým výběrem na poli programovacích jazyků by se možná více hodila volba programovacího jazyka Python, který vychází z rodiny programovacích jazyků C, C++ a C#, ale dle mého názoru je jednodušší, přehlednější, HW méně náročný, v OS Linux a MacOS je nativně přítomný a je zdarma.

Gymnázium Jana Blahoslava, Přerov

CHARAKTERISTIKA ŠKOLY

Gymnázium Jana Blahoslava (oficiální název zní: Gymnázium Jana Blahoslava a Střední pedagogická škola, Denisova 3, Přerov, avšak do výzkumu bylo zařazeno a analyzováno pouze samotné gymnázium a jeho vzdělávací činnost) v Přerově je známo jako gymnázium primárně zaměřené na výuku cizích jazyků. Škola poskytuje budoucím žákům 4letý obecný vzdělávací cyklus nebo 6letý cyklus s rozšířenou výukou jazyka. Samotné vybavení školy digitálními technologiemi pro výuku žáků je v dnešní době lehce nadprůměrné, vybavení pro výuku ICT spíše průměrné. Škola není nikterak speciálně zaměřena technickým směrem (ne, že by výuka technicky zaměřených předmětů neprobíhala) a prezentuje sama sebe hlavně na poli jazykových dovedností. Škola poskytuje svým studentům přípravu a následnou možnost získání jazykových certifikátu.

VÝUKA PROGRAMOVÁNÍ DŘÍVE

Na škole dříve programování a jeho výuka v lehké formě probíhaly. Žáci programovali v jazyku PASCAL (v jisté době bylo k dispozici i jeho rozšíření TurboPascal) a tvořili v něm jednoduché úlohy.

ANALÝZA ŠVP, ICT A PROGRAMOVÁNÍ DNES

Výuka ICT je na gymnáziu Jana Blahoslava mírně podhodnocena. Ročníky *„nižšího gymnázia“* (8. a 9. ročník ZŠ) mají přiděleny 2 hodiny týdně. Na vyšším gymnáziu se žáci k ICT dostanou v prvním a druhém ročníku se stejnou hodinovou dotací, ve 3. ročníku výuka ICT neprobíhá, ve čtvrtém ročníku je to pak 1 hodina týdně. Žáci si také mohou přibrat seminář ICT jakožto volitelný předmět s hodinovou dotací 2 hodiny týdně, avšak až v posledním roce své vzdělávací cesty. Malý zájem o tento předmět však způsobuje, že se většinou předmět neotevře, a jeho výuka tak neprobíhá. Programování probíhá ve 2. ročníku, a to formou tvorby webových stránek v HTML (programování v HTML však nebylo do výzkumu zahrnuto) a v posledním, tedy 4. ročníku, kdy se programuje v programovacím jazyku Karel a MS Visual Studio.

PLÁNY ŠKOLY NA ZMĚNU VÝUKY PROGRAMOVÁNÍ DO BUDOUCNA

Změny v přístupu k programování nebo změna výuky ICT se oficiálně konat nebudou, škola však chce změny provádět průběžně s ohledem na nynější trendy a vzdělávat žáky podle aktuálních potřeb a nenechat tak žáky zaostávat.

MATURITNÍ ZKOUŠKA

Pokud se otevírá seminář z ICT a učitel vyjde žákům vstříc, je možné konat maturitní zkoušku z toho předmětu, žáci pak musí prokázat znalosti z teorie a prakticky naprogramovat jednoduchou úlohu v jistém programovacích jazyku.

Poznámky autora

Škola je primárně zaměřena jazykovým směrem, i přes to by výuka ICT mohla být na této škole brána víc vážně. Autor je zároveň i bývalým absolventem této školy a v době studia byla výuka ICT vedena, na tehdejší poměry, velice podprůměrně. Od jisté doby uplynulo pár změn a změnila se i hodinová dotace, kdy se přidala jedna hodina výuky za týden pro 4. ročník. Pokud ale mají absolventi získat kvalitní vzděláni a znalosti i z oblasti ICT, změny by měly být větší a lepší, a to i přes to, že je škola primárně zaměřena na výuku cizích jazyků.

Slovanské Gymnázium, Olomouc

CHARAKTERISTIKA ŠKOLY

Slovanské Gymnázium v Olomouci nabízí potencionálním žákům možnost studovat obecný 8letý vzdělávací cyklus, 4letý obecný vzdělávací cyklus a mimo to ještě 6letý vzdělávací cyklus zaměřený na výuku jazyků, tzv. česko-francouzskou sekci. Oblast ICT nalezneme úplně ve všech těchto vzdělávacích programech, programování samotné přímo v ŠVP pak také, ale jenom ve velmi omezené míře. Škola se pyšní tím, že v posledních letech prošla rekonstrukcí a v nové budově (dostavbě) tak vzniklo nové místo pro laboratoře a speciální pracoviště, mimo jiné i učebny vybavené pro výuku ICT.

VÝUKA PROGRAMOVÁNÍ DŘÍVE

Před příchodem RVP a povinného zřizování ŠVP do každé školy, se na škole programovalo v programovacím jazyku PASCAL. Programování v PASCALu však bylo určeno pouze žákům, kteří si vybrali *„seminář ICT“*. Otevření tohoto semináře stejně jako u výše zmíněného gymnázia záviselo na zájmu studentů, kdy se kolikrát předmět ani nemusel otevřít z důvodu malého zájmu. Občas se stalo, že se do semináře zapsal 1 nebo 2 žáci a záleželo tedy jenom na vyučujícím, jestli žákům vyjde vstříc i přes malý zájem (většinou byl ale učitel informatiky rád, že mají někteří žáci zájem a předmět se otevřel, v tomto případě tito žáci většinou z předmětu informatiky a programování i maturovali).

ANALÝZA ŠVP, ICT A PROGRAMOVÁNÍ DNES

Na gymnáziu se žáci s předmětem informatiky seznamují již od primy. Žáci se tedy v oblasti ICT vzdělávají již od nástupu do školy. V hodinách ICT standardně probírají HW, SW, internet, multimédia, bezpečnosti při práci s PC apod. Samotné programování se zařazuje do výuky až v sextě (2. ročník SŠ) a to při hodinové dotaci 2 hodiny týdně. Žáci probírají problematiku programování skrze předmět *„informatiky“*, který spadá pod celek ICT, programují skrze kancelářský balík Microsoft Excel ve Visual Basic Excel – rozšíření vývojáře. V sextě pak oficiálně výuka informatiky na gymnáziu končí. Pokud by chtěli žáci probírat další témata z oblasti ICT více do hloubky, mají možnost výběru semináře ICT, kde se také mimo běžnou výuku teorie programuje nadále v VB Excel. Výuku pokročilého programování v semináři ICT zajišťuje externí učitel z přírodovědecké fakulty univerzity Palackého z katedry informatiky.

MATURITA

Na škole je možné provést maturitní zkoušku z programování. Ke zkoušce jsou připuštěni pouze žáci, kteří si vybrali předmět semináře ICT a prokázali tak předešlou znalost jak z teoretické části oblasti ICT, tak z praktického programování. Maturitní zkouška probíhá tak, že si žák vybere 1 teoretickou otázku z oblasti ICT a druhá část je praktický úkol z programování. Pro úspěšné složení

maturitní zkoušky stačí žákovi pouze polovina, tj. buď teoretická, nebo praktická část. Většinou však žáci, kteří si seminář ICT vybrali a chtějí maturovat z ICT, zvládnou zpravidla úkoly oba – dokazují to výborné výsledky těchto žáků z maturitních zkoušek.

PLÁNY ŠKOLY NA ZMĚNU VÝUKY PROGRAMOVÁNÍ DO BUDOUCNA

Do budoucna škola počítá s tím, že by v rámci Strategie digitálního vzdělávání do roku 2020 mohla programování zařadit již od primy a sekundy (6. a 7. ročník ZŠ), a to skrze rozhraní LEGO Mindstorms, ve vyšších ročnících pak pokročilé programování v jazyku C. Škola je na tuto změnu, co se týče vybavení, připravena. Vyřešit se však musí problém s rekvalifikací učitelů, kdy na škole nepůsobí informatici přímo vystudovaní, ale většinou výuku ICT zajišťují učitele matematiky či fyziky, kteří si oblast ICT dostudovali, mají ICT jako hobby nebo učí neaprobovaně.

Poznámky autora

Výuka semináře ICT je vedena odborníkem externistou z PřF UP, žáci tak mají výuku zprostředkovanou odborníkem, který nejen, že umí programovat, ale navíc se pohybuje na akademické půdě, a má tak možnost předávat nejnovější informace, se kterými přijde do styku on sám. Může také žákům dávat informace a cenné rady z oblasti ICT, kdy si je vědom chyb vysokoškolských studentů a žáky gymnázia na tyto chyby může upozornit.

Gymnázium Čajkovského 9, Olomouc

CHARAKTERISTIKA ŠKOLY

Gymnázium Čajkovského 9 v Olomouci, nabízí studentům 8letý vzdělávací cyklus, který není přímo profilován humanitně, sportovně či přírodovědecky. Dále 6leté vzdělávání, které je zaměřeno na výuku jazyků a 4letý cyklus zaměřený sportovním směrem nebo na rozšířenou výuku německého či španělského jazyka. Škola působí neutrálním dojmem, poskytuje učební prostory pro výuku cizích jazyků a prostory pro sportovní aktivity, jako je tělocvična a sportovní hala. Vybavení učebny pro výuku ICT se jeví na dnešní dobu podprůměrně.

VÝUKA PROGRAMOVÁNÍ DŘÍVE

Před zavedením RVP a povinností vytvářet ŠVP škola vedla předmět informatiky a učilo se i samotné programování. Programovalo se v programovacím jazyce PASCAL. Programování však bylo umožněno pouze studentům, kteří si vybrali předmět *„seminář ICT“*. Možnosti programování byly na škole velice omezené, a to z důvodu tehdejšího vybavení školy, přítomnosti pouze jednoho učitele programování na škole a malého zájmu žáků o tuto problematiku.

ANALÝZA ŠVP, ICT A PROGRAMOVÁNÍ DNES

Oblast informatiky se nyní na škole vyučuje, avšak žák, který na školu nastoupí, se s předmětem *„informatiky“* nedostane do kontaktu ihned. Podle ŠVP začíná výuka *„informatiky“* až v tercii (8. ročník ZŠ a ekvivalent 6letého cyklu), kdy se žáci postupně standardně učí HW, SW, oblast internetu, kancelářský balík Microsoft Office, bezpečnou práci s PC apod. Výuka ICT probíhá až do sexty (2. ročník SŠ a ekvivalent 6letého cyklu), po sextě však výuka ICT končí. Oblast programování se *„fyzicky“* na této škole vůbec nevyučuje, po rozhovoru s učiteli ICT bylo zjištěno, že výuka programování je vedena skrze projekt, kdy se žáci programování učí na pedagogické fakultě UP Olomouc.

PLÁNY ŠKOLY NA ZMĚNU VÝUKY PROGRAMOVÁNÍ DO BUDOUCNA

Do budoucna škola počítá s tím, že by mělo dojít ke změně jak v přístupu výuky samotného předmětu ICT, tak i programování. Učitelé ICT již teď navštěvují různá školení zaměřená na programování a jeho výuku, nicméně na otázku: *„Jak by měla výuka programování probíhat a jaký programovací jazyk by k tomu měl být využitý?“* se odpověď nepovedlo získat. Škola si je vědoma, že významnost oblasti ICT v dnešní době narůstá čím dál tím více, a že je důležité změny provést, ale zatím je všechno ve fázi *„plánování/spekulování“,* jak uvedl respondent z této školy.

Poznámky autora

Škola je v oblasti výuky ICT (oproti jiným školám) značně pozadu, časové dotace pro předmět ICT jsou nízké a žáci se k předmětu *„informatiky“* dostávají, vzhledem k dnešní době, dost pozdě. Strategie digitálního vzdělávání do roku 2020 předpokládá, že školy začnou oblast ICT vyučovat efektivněji a začlení i samotné programování. Většina jiných škol již nějakou představu o budoucím programováním má, tato škola si však dává s rozhodnutím značně *„načas“* a jeví se jistým způsobem nepřipraveně.

Střední průmyslová škola strojnická, 17. listopadu, Olomouc

CHARAKTERISTIKA ŠKOLY

Tato škola je zaměřená technickým směrem, mezi učebními obory najdeme strojírenství se zaměřením na počítačovou podporu konstruování, počítačovou podporu výroby, management jakosti a průmyslový design, dále obor zpracování usní, platů a pryže se zaměřením na zpracování platů a obor provozní technika. Škola se pyšní 96% úspěšností žáků v denním studiu u státních maturitních zkoušek. Vede filosofii, že technika je oborem budoucnosti a absolventi školy vždy měli, mají a budou mít uplatnění. Mimo technické vzdělávání škola poskytuje možnosti jazykového vzdělávání a mnoho sportovních mimoškolních aktivit. Škola disponuje dílnami pro výuku strojírenských oborů, vybavení pro výuku informatiky (popř. programování) je nyní na lehce nadprůměrné úrovni.

VÝUKA PROGRAMOVÁNÍ DŘÍVE

Výuka programování na této škole probíhal i dříve, avšak ne v tradičním pojetí a přímo na stolním počítači. Na škole se vyučovalo programování pro CNC stroje v rámci *„odborného vzdělávání“*. Žáci tedy přímo nebyli seznámeni s klasickým způsobem programování, nýbrž spíše s určitou formou paradigmatu declarative-first, kdy si však mimo získané znalosti programování CNC stroje odnesli i způsob uvažování v jiné rovině a nahlížení na jisté problémy, jako na matematickou úlohu, již lze vyřešit *„určitým vzorcem“*.

ANALÝZA ŠVP, ICT A PROGRAMOVÁNÍ DNES

Škola je zaměřená technicky, nicméně programování v běžných programovacích jazycích zde zastoupeno není. Žáci mají ve všech ŠVP předmět informatiky, která obsahuje učivo: ovládání kancelářského balíku Microsoft Office, užívání počítače (HW, SW, bezpečnost při práci s PC), internet a multimédia. Programování zde ale nalézt můžeme, žáci se učí programovat CNC stroje, a to v rámci celku *„odborné vzdělávání“.* Mimo programování CNC strojů se žáci učí pracovat i s CAD a CAM programy, kde probíhá modelování a konstrukce součástek technického charakteru – průmyslový design.

PLÁNY ŠKOLY NA ZMĚNU VÝUKY PROGRAMOVÁNÍ DO BUDOUCNA

V rámci Strategie digitálního vzdělávání do roku 2020 škola počítá s tím, že se do ŠVP přidá možnost programování a algoritmizace skrze stavebnice LEGO Mindstorms. Programování skrze tyto stavebnice by však mělo být zařazeno jenom jako doplnění vzdělávání, přičemž zaměření této školy není přímo na informatiku, a proto by programování mělo sloužit pouze jako rozšíření žákových dosud získaných vědomostí. Programování by mělo být zahrnuto do předmětu *„informatika“* pouze pro 3. ročník. Pro *„informatiku“* jsou nyní celkově vytýčeny 3 hodiny v týdnu,

přičemž samotné programování skrze LEGO Mindstorms by mělo být dotováno maximálně 2 hodinami v týdnu.

Poznámky autora

Škola není zaměřena na výuku informatiky, i přes to chce ale držet krok s dobou a výuku programování poskytnout žákům v budoucnu formou algorithms-first, pomocí stavebnic LEGO Mindstorms. Tento krok školy je pro budoucí žáky jistě přínosem, programování neučí pouze programovat, ale zároveň vidět a chápat problémy a jejich řešení v jiné rovině, což se nehodí jenom lidem studující technické obory, ale obecně každému, kdo žije v dnešní digitální společnosti.

Střední škola Technická a obchodní, Kosínova 4, Olomouc

CHARAKTERISTIKA ŠKOLY

Škola byla původně založena jako učňovská škola v roce 1945. Od té doby se hodně věcí změnilo a škola mimo možnost získání výučního listu z různých oborů nabízí i úplné středoškolské vzdělání a v rámci doplnění kvalifikace i celoživotní formy vzdělávání (nástavbové vzdělání v oboru podnikání). Největší poptávka na škole je po elektrotechnickém a strojírenském oboru – v těch má škola co nabídnout díky dlouholeté tradici a zkušenostem v těchto oborech. Mimo standardní vyučovací prostory škola nabízí i výuku v elektro-laboratořích, dílnách a prostorech pro odborné vyučování vybavené patřičným vybavením. Škola také nabízí propustnost učňovských oborů společně s maturitními – pokud se tedy žákovi nedaří, může přestoupit z 4letého maturitního oboru na 3letý učňovský, škola tak vychází vstříc svým žákům a snaží se jim pomoci v každém ohledu.

VÝUKA PROGRAMOVÁNÍ DŘÍVE

Před příchodem RVP a ŠVP se na škole v osnovách programování vyskytovalo. Programovali však pouze žáci, kteří měli zaměření na telekomunikace a žáci elektrotechniky, kteří se učili programovat jednoduché příkazy v programovacím jazyku ASSEMBLER.

ANALÝZA ŠVP, ICT A PROGRAMOVÁNÍ DNES

S příchodem RVP a ŠVP byl do osnov v rámci ICT zařazen předmět *„informatiky“*. Informatiku mají na škole všichni studenti od 1. ročníku. Každý žák si projde standardním úvodem učiva informatiky jako je: bezpečná práce s počítačem, internet, multimédia, kancelářský balík Microsoft Office, HW a SW. Od druhého ročníku se však žáci na oborech telekomunikace a elektrotechnik seznamují s programovacím jazykem C a výuka v něm probíhá až do 3. ročníku. Obor elektrotechnik pak ve 4. ročníku programování nemá. Obor telekomunikace pak ve 4. ročníku přechází z programovacího jazyka C na platformu Arduino, kde se za pomocí jednodeskového počítače programují jednoduché programy pro řízení různých elektrotechnických zařízení. Výuka ASSEBLERU na škole stále probíhá, nicméně je postupně na útlumu (nižší programovací jazyk = problémy v abstrakci, navíc učitel předmětu je pomalu na odchodu do důchodu a programování obecně v nižších programovacích jazycích není moc oblíbené).

MATURITNÍ ZKOUŠKA

Na škole je možné maturovat z předmětu *„informatika“*. Pro to, aby žák úspěšně prošel maturitní zkouškou z ICT, musí umět prakticky naprogramovat jednoduchou zadanou úlohu, předvést její funkčnost a okomentovat části kódu. Maturitní zkouška probíhá z programovacího jazyka C a na platformě Arduino.

PLÁNY ŠKOLY NA ZMĚNU VÝUKY PROGRAMOVÁNÍ DO BUDOUCNA

V rámci Strategie digitálního vzdělávání do roku 2020 škola nepočítá s většími změnami výuky ICT a jinou formu programování či přerozdělení nebo navýšení kapacity hodin ICT. Škola nebere Strategie digitálního vzdělávání do roku 2020 jako směrnici a sama si určuje trend, čas a směr vylepšování svého prostředí (dílny, pracoviště a laboratoře), kdy se snaží v těchto ohledech držet krok s dobou nepřetržitě. Změny v ICT se snaží reflektovat průběžně, což se škole v podstatě daří.

Střední průmyslová škola Brno, Purkyňova

CHARAKTERISTIKA ŠKOLY

Střední škola Purkyňova v Brně je škola zaměřená technickým směrem. I přes známý fakt, že technické obory jsou mezi žáky v ČR méně oblíbené a žáky pro studium těchto oborů se většinou školy snaží nahnat *„kde se dá“*, jde tato škola opačným směrem, kdy si své studenty vybírá pomocí přijímacího řízení. Škola nabízí 10 maturitních oborů, nicméně obor informačních technologií zaujímá první místo a tento obor má vyhrazenou nejvyšší kapacitu. Ačkoliv škola působí *„staře“* a budova je stavěna ve stylu kubismu-funkcionalismu, uvnitř poskytuje žákům moderní technické prostory pro kvalitní výuku, vybavení pro výuku ICT je nadprůměrné. Škola se také pyšní úspěchy nejen na poli informačních technologií a robotiky, ale i ve sportech.

VÝUKA PROGRAMOVÁNÍ DŘÍVE

Výuka programování na této škole před zavedením RVP a ŠVP fungovala. Programování probíhalo již od 1. ročníku, a to v jazyku C. Nadále se programovalo až do 3. ročníku ve všech oborech, ve 4. ročníku pak programovali pouze žáci profilující se jako budoucí programátoři ve speciálním předmětu *„seminář pro programování“*.

ANALÝZA ŠVP, ICT A PROGRAMOVÁNÍ DNES

Dnes probíhá výuka programování podobně jen s mírnými obměnami. Programování stále probíhá již od 1. ročníku ve všech oborech, ale žáci se seznamují s programováním skrze *„algorithms-first“* způsob, kdy škola spolupracuje s MUNI a studenti programují v *„želví grafice“* [18], postupně však přechází na Python a programují v něm pomocí Imperative-first paradigmatu. V pythonu žáci zůstávají, ale po roce studia se přesunují od základů k OOP, 3. ročník je vyhrazen PHP, databázím, Javascriptu, webovým aplikacím a budoucí programátoři (stejný případ jako *„výuka programování dříve“*) pak i Javu. Ve 4. ročníku programují pouze programátoři. Časová dotace (počet hodin na jeden týden) je: 3 hodiny pro 1. ročník, 2 hodiny pro 2. ročník, 3. ročník 3 hodiny (s výjimkou programátorů, kteří mají 6 hodin) a 4. ročník pouze pro programátory s 6 hodinami.

MATURITNÍ ZKOUŠKA

Maturitní zkouška je na této škole v podstatě samozřejmostí. Studenti musí projít teoretickou i praktickou zkouškou. Teoretická zkouška zahrnuje soubor obecných znalostí, praktická

[18] Želví grafika – zjednodušené prostředí programovacího prostředí LOGO, kdy žák ovládá pomocí jednoduchých příkazů želvu a její pohyb po mapě. Založeno na Algorithm-first, kdy se žák učí vštípit si zvyky programování a nutnost určitých pravidel při programování a syntaxi.

pak programování, kdy žáci mohou používat dokumentaci a internet – výsledek pak musí předvést a okomentovat části kódu. Posledních 8 let také na škole funguje možnost maturity z programování formou dlouhodobého projektu, kdy žáci nemusí programovat na místě jednoduchý náhodný úkol, ale předvádí svůj vlastní *„velký“* projekt, který obhajují.

PLÁNY ŠKOLY NA ZMĚNU VÝUKY PROGRAMOVÁNÍ DO BUDOUCNA

Škola zatím do budoucna neplánuje větší změny, škola chce pouze dosáhnout větší spolupráce s MUNI, vytvořit užší spojení a tvořit společné projekty.

Poznámky autora

Škola neplánuje větší změny do budoucna, ale v tomto případě to ani není nutné. Škola se jeví velice schopnou v přípravě svých žáků na budoucí povolání, či další studium na VŠ. Škola navíc přechodem od programovacího jazyku C na Python již v dřívější době *„splnila povinnost“* sledovat aktuální trendy a *„neusnout na vavřínech“*.

Střední průmyslová škola strojní a elektrotechnická, Dukelská 13, České Budějovice

CHARAKTERISTIKA ŠKOLY

Škola zaměřená technickým směrem s dlouhodobou tradicí, vznikla již v roce 1913 jako německá odborná škola kovorobná. Po dlouhá léta poskytovala technické vzdělání lidem v oblasti jižních Čech a v současnosti nabízí 2 obory – obor elektrotechniky a obor strojírenství, oba jsou maturitní. Škola mimo klasické prostory pro výuku žáků disponuje dílnami pro výuku odborných předmětů, počítačovou učebnou, laboratořemi, velkou tělocvičnu a posluchárnou.

VÝUKA PROGRAMOVÁNÍ DŘÍVE

Výuka programování na této škole v minulosti před nástupem RVP a ŠVP probíhala. Výuka probíhala v jazyce PASCAL a v rámci odborného vzdělávání probíhalo i programování CNC strojů

ANALÝZA ŠVP, ICT A PROGRAMOVÁNÍ DNES

Žáci se dnes, za dob RVP a ŠVP, k programování dostanou také, a to skrze předmět informatiky v oblasti ICT. Žáci při příchodu na školu v 1. ročníku proberou teorii z oblasti ICT, ve 2. ročníku už se v teoretické rovině dostanou např. k databázím. Samotné programování čeká na žáky ve 3. a 4. ročníku, kdy programují nejdříve v HTML a následně v jazyku C. Čtvrtý ročník je věnován z půlky počítačovým sítím (+ její praktická realizace) a tvorba webu za využití pokročilejších technologií, jako je třeba JavaScript, PHP a MySQL. Hodinová dotace je rozdělena celkem rovnoměrně, kdy činí 2 hodiny týdně po celou dobu studia, tedy 4 let.

MATURITNÍ ZKOUŠKA

Maturitní zkouška z předmětu ICT probíhá standardním způsobem, kdy žáci musí předvést své teoretické znalosti z oblasti ICT a naprogramovat jednoduchou zadanou úlohu.

PLÁNY ŠKOLY NA ZMĚNU VÝUKY PROGRAMOVÁNÍ DO BUDOUCNA

Škola zatím neplánuje větší změny ve výuce ICT nebo v přístupu k výuce programování.

Poznámky autora

Škola sice žádné změny v oblasti programování nechystá, zde to ale není potřeba. Škola primárně vychovává absolventy v oborech elektrotechnika a strojírenství, žáci se v odborných předmětech učí programovat CNC stroje, a k tomu ještě škola poskytuje značně pokročilou výuku programování v jazyce C, i když na informační a komunikační technologie primárně zaměřena není

– škola tak v podstatě sleduje trend nezbytnosti výuky programování, kdy je oproti jiným školám značně napřed.

Gymnázium, Praha 7, Nad Štolou 1

CHARAKTERISTIKA ŠKOLY

Gymnázium Nad Štolou 1 nabízí budoucím uchazečům studium 8leté obecné, 6leté obecné, 4leté obecné a 4leté – sportovně zaměřené. Ačkoliv jsou obory popsány jako *„obecné“* v ŠVP škola uvádí, že se škola profiluje především na výuku cizích jazyku, kdy je povinností angličtina a druhý jazyk je volitelný. Škola si také své budoucí žáky pečlivě vybírá, kdy uchazeči musí podstoupit přijímací zkoušku z českého jazyka, matematiky a speciální školní přijímací zkoušku. Gymnázium se jeví moderní, mimo prostory pro standardní výuku škola disponuje sportovními areály a školní knihovnou.

VÝUKA PROGRAMOVÁNÍ DŘÍVE

Výuka programování na gymnázium probíhala i dříve, než přišlo v platnost RVP a ŠVP. Programování se na škole dříve vyučovalo v předmětech ICT v programovacích jazycích Pascal a Delphi. Programování se dostalo ke studentům v Sextě (2. ročník SŠ a ekvivalentní ročník 6letého cyklu) a pro zájemce i v Oktávě (4. ročník SŠ a ekvivalentní ročník 6letého cyklu) v rámci speciálního předmětu pro budoucí maturanty z předmětu ICT. Maturitní zkouška obsahovala teoretickou a praktickou část, kdy praktická část obsahovala programování jednoduchého úkolu.

ANALÝZA ŠVP, ICT A PROGRAMOVÁNÍ DNES

S příchodem RVP a ŠVP se výuka ICT skrze programování mírně změnila. V předmětu ICT se programování dostane ke studentům v kvintě (1. ročník SŠ a ekvivalentní ročník 6letého cyklu) v rámci obsahového celku *„tvorba webových stránek“* v programovacím jazyku Javascript. Žáci pak pokračují v programování nadále v Sextě, kdy přejdou na programovací jazyk Python. V sextě oficiální povinná výuka ICT, a s tím i programování, končí, pro zájemce je však možnost volby semináře ICT, kde se mohou programování věnovat více do hloubky, přičemž se naváže opět na Python, ale žáci programují více sofistikované úlohy.

MATURITNÍ ZKOUŠKA

Maturitní zkouška je určená pouze pro studenty, kteří si zvolili *„seminář ICT“*. Maturitní zkouška se skládá z teoretické části (teoretické znalosti z oblasti ICT) a praktické části, kdy žák obhajuje větší projekt, který musel začít tvořit již dříve.

PLÁNY ŠKOLY NA ZMĚNU VÝUKY PROGRAMOVÁNÍ DO BUDOUCNA

Škola do budoucna zatím žádné větší změny v oblasti výuky ICT a programování nechystá.

Poznámky autora

I když škola žádné větší změny nechystá, není to v tomto ohledu potřeba. S ohledem na to, že se jedná o gymnázium s *„obecným zaměřením“*, kdy profilace školy hovoří o výuce cizích jazyků, je výuka programování vedena dostatečně na to, aby absolvent školy neměl větší problémy s dalším studiem v oblasti ICT a programování.

Střední škola polytechnická, Rooseveltova 79, Olomouc

CHARAKTERISTIKA ŠKOLY

Škola nabízí budoucím žákům vzdělání ve směrech: obchodní, strojírensko-automobilní, dřevařský a stavební. Mimo školních prostor pro standardní výuku škola disponuje sportovním areálem, dílnami a laboratořemi. Škola navíc poskytuje svým žákům v rámci studia možnost získání řidičského průkazu a ubytování ve svých vlastních prostorech/kolejích.

VÝUKA PROGRAMOVÁNÍ DŘÍVE

Před zavedením RVP a ŠVP na škole výuka programování neprobíhala.

ANALÝZA ŠVP, ICT A PROGRAMOVÁNÍ DNES

S nástupem RVP a ŠVP se na škole můžeme setkat s lehkou formou programování a algoritmizace, avšak ve velmi omezené míře. Žáci programují pomocí algorithm-first v programovacím jazyku *„želví grafika“* (viz výše). Na žáky však není vyvíjen *„tlak“*, aby programování zvládli stoprocentně a ve výsledku tak občas probíhá výuka programování pouze informativně. V oblasti ICT se na této škole klade důraz na CAD programy, práci v nich a jejich praktické využití v životě.

MATURITNÍ ZKOUŠKA

V oblasti ICT maturitní zkouška na této škole neprobíhá.

PLÁNY ŠKOLY NA ZMĚNU VÝUKY PROGRAMOVÁNÍ DO BUDOUCNA

Plány školy na změnu v přístupu k programování zatím nejsou. Uvažuje se o vytvoření volitelného předmětu *„semináře ICT“*, kde by mohla probíhat i výuka programování. V jakém programovacím jazyce a s jakou časovou dotací se zatím ale ještě neví.

Poznámky autora

Škola produkuje absolventy z řad stavitelů, řemeslníků, ale i obchodníků a automechaniků. U některých oborů není přímo potřeba podporovat výuku programování (řemeslníci a stavitelé), mechanici a obchodníci by už ale nějaké, alespoň menši, povědomí o programování mít mohli, v tomto ohledu ale výuka ICT a programování v želví grafice nejspíše dostačuje.

6. Vyhodnocení výzkumu

6.1. Sjednocené výsledky

6.1.1. Tabulkový výčet

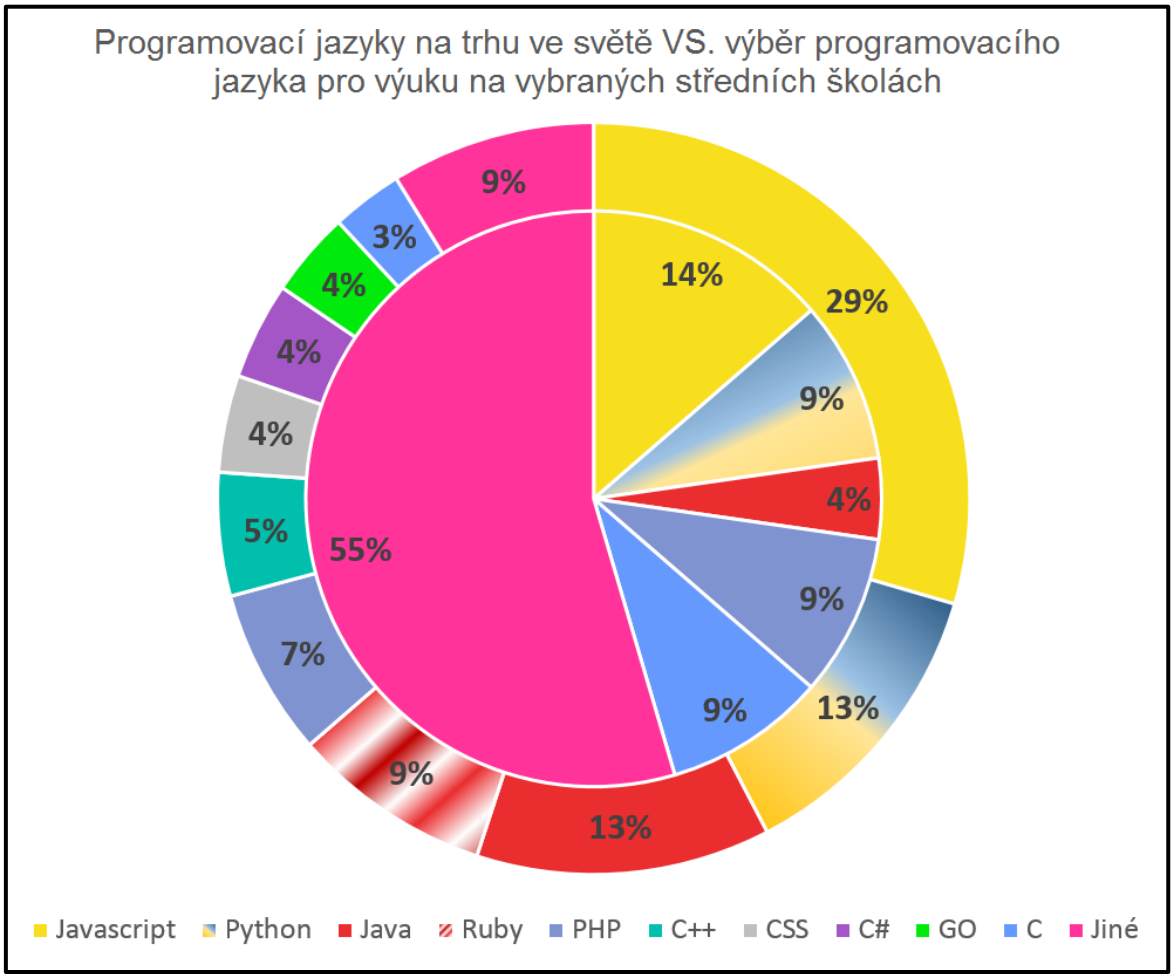
Název školy	Zaměření školy	Programování dříve	Počet hodin ICT				Seminář ICT				Programování nyní	Maturita z programování	Změny do budoucna	Programování "nahrazeno" jinou činností zasahující do ICT / jiné poznámky
			1	2	3	4	1	2	3	4				
Gymnázium Jakuba Škody, Přerov	obecné	Basic, Pascal	2	1	–	–	–	–	2	2	Pascal, LOGO, Basic	Teoretická a praktická	Více jazyků nahradit pouze C#, ten probírat více do hloubky	–
Gymnázium Slovan, Olomouc	obecné, jazykové	Pascal	2	–	–	–	–	–	2	2	VB Excel	Teoretická a praktická	Programovat již od Primy skrze LEGO Mindstorms	Seminář ICT vede externista z PřF UPOL
Gymnázium Čajkovského, Olomouc	obecné, jazykové, sportovní	Pascal	2	2	–	–	–	–	–	–	–	–	Zatím nepřipraveno	Výuka programování skrze projekt s PdF UPOL
Střední průmyslová škola strojnická, 17. listopadu, Olomouc	technické	Programování strojů CNC	1	2	1*	2*	–	–	–	–	*CNC a konstrukční CAD/CAM	–	Programování v LEGO Mindstorms	Programují se pouze stroje CNC, výuka CAD/CAM
Střední škola Technická a obchodní, Kosínova 4, Olomouc	technické	ASSEMBLER	2	2	2	2*	–	–	–	–	ASSEMBLER, C, Arduino	Teoretická a praktická	Škola sleduje trendy v ICT, změny provádí průběžně.	*Programování pouze pro obor telekomunikací
Střední průmyslová škola, Purkyňova, Brno	technické	C	3	2	3	–	–	–	3	6	LOGO, Python, Javascript, Java, PHP, SQL	T+P/obhajoba velkého projektu	Do budoucna posílit projektovou činnost s MUNI	Škola zaměřená primárně na ICT
Gymnázium, Nad Štolou 1, Praha 7	obecné, jazykové, sportovní	Pascal, Delphi	2*	–	–	–	–	–	1	1	Javascript, Python	Teor. část + obhajoba projektu	Škola sleduje trendy v ICT, změny provádí průběžně.	*Pro jazykáře je hodinová dotace snížena na 1h/t
Střední škola polytechnická, Rooseveltova 79, Olomouc	technické, výuční	–	1	1	1*	1	–	–	–	–	*Želví grafika, velice omezeně	–	Uvažuje se o vytvoření nepovinného sem. ICT pro obory s maturitou	Programování pouze v maturitních oborech
Gymnázium Jana Blahoslava, Přerov	obecné, jazykové	Pascal	2	2	–	1	–	–	–	2	Karel, MS Visual Studio	Teoretická a praktická	Snaha změny reflektovat průběžně	–
Střední průmyslová škola strojní a elektrotechnická, Dukelská 13, České Budějovice	technické	Pascal, programování strojů CNC, C	2	2	2	2	–	–	–	–	C, PHP, JavaScript, MySQL	Teoretická a praktická	Škola sleduje trendy v ICT, změny provádí průběžně.	Výuka programování CNC neomezuje výuku klasického programování

Tabulka 1: Porovnání jednotlivých středních škol

Z výše vyobrazené tabulky lze vyčíst výsledky výzkumu v *„číslech“* a také porovnat jednotlivé školy a jejich výuku ICT včetně programování. **MODŘE** byly zvýrazněny školy, které samy sebe prezentují jako technické, **ŽLUTÉ** zvýraznění značí atypickou situaci programování (např. programování mimo běžný stolní počítač nebo běžné programovací prostředí), **FIALOVÉ** zvýraznění značí výjimku v programování, a to sice programování v nižším programovacím jazyce. **TMAVĚ**

ZELENOU barvou jsou zvýrazněny ročníky, kde probíhá výuka programování a algoritmizace (mezi programování a algoritmizaci nebyla zahrnuta výuka tvorby webových stránek za použití planého HTML), **SVĚTLE ZELENOU** barvou je označena skutečnost, že se škola do budoucna nějakým způsobem snaží o pozitivní změny ve způsobu výuky ICT a podporu programování, **ČERVENÁ** naopak značí, že změny nejspíše škola neprovede, **ŠEDÁ BARVA** nakonec značí poznámku k samotné výuce programování.

6.1.2. Grafický výčet



Obrázek 15: Graf znázorňující programovací jazyky na trhu ve světě a jazyky užité k výuce na zkoumaných školách.

Koláčový graf znázorňuje procentuální podíl využití programovacích jazyků na trhu ve světě (vnější kruh) a využití programovacích jazyků ve výuce na zkoumaných školách (vnitřní kruh). Graf je doplněn o legendu, která sedí na obě části grafu – jednotlivé programovací jazyky jdou po sobě ve stejném pořadí, výřezy jsou pro snadnější orientaci doplněny o procentuální podíl užití jistého programovacího jazyka (samotná tvorba webu pomocí planého HTML zde není zahrnuta jako výuka programování a algoritmizace, proto v grafu tyto informace obsaženy nejsou).

Je možné vypozorovat, že položka *„Jiné“* vyplňuje ve světovém měřítku pouze 9 %, ale na školách tvoří více, než polovinu – je to způsobené tím, že školy leckdy používají pro výuku programování i pseudo-programovací jazyky, které se v běžné praxi programování, mimo zmíněnou

výuku, nepoužívají. Dále je položka *„Jiné“* ve školním prostředí doplněna o některé jazyky, které nejsou na trhu tak oblíbené, v tomto případě jde o MS Visual Studio, LOGO, VB Excel, MySQL, SQL, Karel a ASSEMBLER.

6.2. VÝČET VÝZKUMNÝCH PŘEDPOKLADŮ

Aspoň na 70 % všech dotazovaných škol již dříve programování probíhalo.

Z výše uvedené tabulky můžeme vyčíst údaje, které nám sdělují, že programování již dříve probíhalo na 9 z 10 zmiňovaných škol. V tomto případě programování dříve probíhalo na 90 % dotazovaných škol, čímž tento výzkumný **předpoklad můžeme považovat za platný/splněný.**

Aspoň na 70 % všech dotazovaných škol se mimo výuku ICT také nyní programuje.

Opět můžeme z tabulky vyčíst, že nyní programování probíhá na 9 z 10 dotázaných škol. V tomto případě tedy probíhá programování na 90 % škol, proto můžeme **výzkumný předpoklad považovat za platný/splněný.**

100 % škol, které jsou ryze technické, vyučovalo programování před zavedením RVP.

Tento výzkumný předpoklad předpokládá, že všechny technicky zaměřené školy prováděly výuku programování i před zavedením RVP. V tabulce se nachází pouze jedna škola, která před zavedením RVP programování nevyučovala, proto tento **výzkumný předpoklad není splněn.**

100 % škol, které jsou ryze technické, dnes vyučuje programování.

Výzkumný předpoklad předpokládá, že po zavedení RVP a nutnosti vytvářet školní vzdělávací plány dojde na každé škole ke změnám, které aspoň nějakým způsobem *„donutí“* školy zavést v jisté míře programování do hodin ICT. Z tabulky lze vyčíst, že všechny dotázané školy v rámci ICT programování vyučují. **Výzkumný předpoklad je tedy platný/splněný.**

Všechny dotázané školy jsou již seznámeny s plánem Strategie digitálního vzdělávání do roku 2020 a mají zpracovaný plán na změnu v přístupu k výuce programování.

Tento výzkumný předpoklad nemá za cíl pouze zjistit, jestli jsou školy na tuto strategii připraveny. Vedlejším účinkem je také možnost zjistit, zda se škola snaží o kvalitní výuku ICT, i když je to občas předmět vnímaný, jako *„neužitečný“* nebo *„odpočinkový“* a jestli si je vědoma, že výuka ICT by se neměla brát na lehkou váhu.

Obrázek 16: Znázornění VP č. 5 v grafu

Z tabulky vyplývá, že 5 škol je s plánem seznámeno a aktivně pracuje na budoucím rozvoji výuky ICT, dalších 5 škol změnu nepovažuje za důležitou, přičemž 4 z těchto škol se snaží reflektovat změny v reálném čase – změna tak není přímo

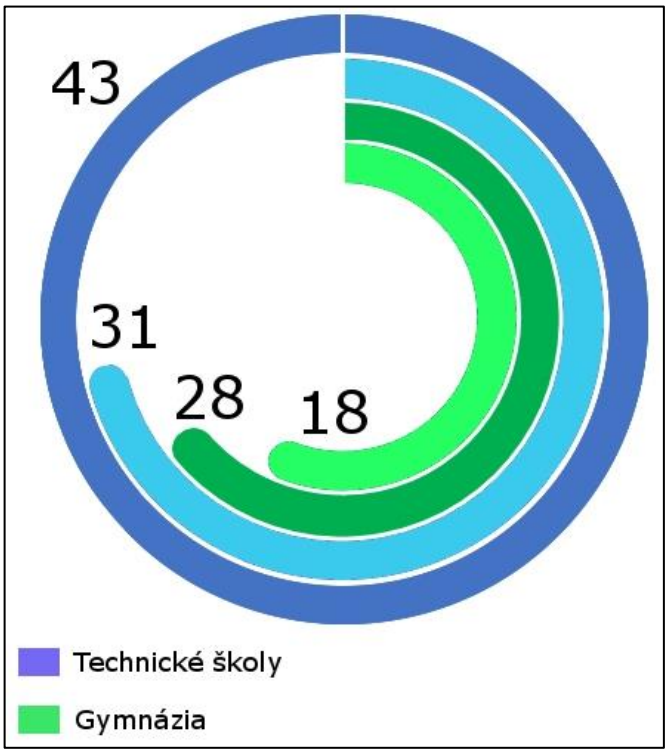
potřeba a absolventi školy by neměli mít větší problémy s uplatněním, vzhledem ke svému získanému vzdělání, nebo jsou dostatečně připraveni na další studium, kde je znalost programování nutností. Jena škola je sice seznámena s plánem, ale vlastní změny zatím neplánuje. **Výzkumný předpoklad tak lze pokládat za nesplněný.**

Poměr výuky programování v rámci ICT se na technických školách a gymnáziích liší.

Výzkumný předpoklad má za cíl zjistit, zda technické školy považují programování za důležité, a proto svým absolventům v tomto předmětu poskytují daleko lepší zázemí než gymnázia. Předpokládali jsme, že technické školy dbají na výuku programování více než gymnázia. Protože je ale pojem *„dbá“* poměrně abstraktní, rozhodli jsme se pro výše uvedenou formulaci výzkumného předpokladu, která je měřitelná. Výzkumný předpoklad tak lze jednoznačně na základě sebraných dat označit za splněný nebo nesplněný.

V tomto předpokladu také musíme opomenout sympatie, či antipatie žáků vůči předmětu ICT, a to, jestli si budou, nebo právě nebudou chtít přidat volitelný předmět semináře ICT mezi své studované předměty. Zkoumaným prvkem bude pouze to, jestli škola lepší podmínky poskytuje, tedy jestli žákům nabízí vyšší hodinovou dotací výuky ICT a programování, bez ohledu na to, zda jde o povinné nebo volitelné předměty. V tomto případě budou sečteny maximální možné počty hodin předmětu *„informatiky“* a jejich seminářů dohromady za všechny technické školy i gymnázia.

Hybridní koláčový graf obsahuje jednu úplnou kružnici a 3 neúplné. Modrá barva a její světlejší odstín označují školy technické, zelená barva a její světlejší odstín pak gymnázia. U grafového znázornění jsou také přítomny hodnoty hodinových dotací pro předmět ICT. Tmavě modrá kružnice znázorňuje pomyslných 100 %, a to proto, že hodinová dotace předmětu informatiky u technických škol dohromady čítala 43 hodin týdně, což je maximální *„naměřená“* hodnota. Hodiny, které byly spjaty s programováním, pak znázorňuje slabě modrá barva, kdy výuka obsahuje i programování. To samé platí pro gymnázia, kdy tmavě zelená barva čítá hodinovou dotaci pro předmět oblasti ICT a světle zelená pak znázorňuje počet hodin, kde se výuka dotýká i programování.



Obrázek 17: Poměr ICT bez programování vůči hodinám s programováním.

Z výše uvedeného grafu lze vyčíst, že školy, zaměřené technickým směrem, opravdu poskytují žákům více hodin pro předmět ICT. Na druhou stranu gymnázia sice neposkytují žákům tak velkou hodinovou dotaci, ale oblast programování se zde objevuje ve větším poměru vzhledem k celkové hodinové dotaci. Proto lze říci, že školy zaměřené technickým směrem sice mají vyšší počet hodin pro předmět ICT, ale jestli dbají na výuku programování více, než gymnázia průkazné není. Poměr výuky programování vůči celkové výuce ICT je vyšší na gymnáziích než na technických školách. **Výzkumný předpoklad lze označit za splněný, poměr výuky v rámci ICT se na technických školách a gymnáziích liší.**

7. Diskuze

7.1. Výsledky výzkumu

V krátkosti lze říci, že výzkum nedopadl až tak špatně. Školy v minulých letech dostaly několikrát dotace na vylepšení svého technického a digitálního arzenálu pro výuku ICT (MŠMT, 2009). Dostatečná vybavenost školy pro výuku ICT je zajisté důležitým faktorem pro samotný proces této výuky. Co se týče Strategie digitálního vzdělávání do roku 2020 se zdá být dobrým spouštěčem (nebo v závěrečné fázi tak trochu drábem), který upozorňuje školy na nutnost změn v koncepci výuky ICT a nutnosti zařadit do ní i samotné programování, jehož znalost bude v příštích letech zajisté nutností. Dotázané školy (a určitě nejen ty) ukázaly, že o této skutečnosti vědí a změny již průběžně provádí, nebo k nim již ve většině případech strategii naplánovanou mají. Najdou se však i takové školy, které změnu nejspíše nechystají, nebo si se změnou *„dávají dost na čas“*. Snad bude těchto škol v ČR minimum a s časem budou ubývat (ať už tím, že zaniknou, nebo výuku ICT změní).

7.2. Návrh na změnu

Z obecného povědomí, i ze získaných informací lze zjistit, že programování je velice *„nestabilní“* a v čase se stále proměňuje, vylepšuje a posunuje stále vpřed. Školy na to nahlíží podobně a snaží se, ve většině případů, svým žákům poskytovat vzdělání takovým způsobem, aby se žáci v budoucnu uplatnili. Nicméně oblast informatiky na školách je stále na špatné úrovni a její výuka by se měla jistým způsobem obměnit např. přerozdělením různých úkonů, tím, že se v českém jazyce může vyučovat MS Word, v matematice MS Excel, výtvarná a estetická výchova může přijmout z ICT možnost vytváření klipů, krátkých žákovských filmů, jejich střih apod. Tyto změny a přerozdělení učiva, které přímo souvisí s výukou ICT, pak celkově urychlí a zlepší možnosti užití počítače pro různé úkony a předmětům IVT/ICT/IK na středních školách dají více času pro samotnou výuku programování (postupně by bylo možné takto postupovat i na školách základních). Výuka programování však nemusí probíhat formou *„opisuj z tabule/plátna“*, nýbrž formou hry, kdy si žák učivo lépe zapamatuje. Když si žák programování užije jako hru a líbivý zážitek, podpoří to celkový proces učení. Jak již bylo v práci zmíněno, programování a jeho výuka nemá a nemusí mít za cíl vytvářet nové programátory, kterých je mimochodem velmi málo a o důležitosti prezence kvalifikovaných lidí pro tyto pozice není pochyb, ale v prvé řadě naučit děti, budoucí generaci, myslet v jiné rovině, dát jim možnost vidět problém, ale hlavně i jeho řešení, v různých rovinách (Paulenková, 2017).

Je obecně známo, že je programování nelehký úkol a je zapotřebí mnoho úsilí nejen tuto činnost vyučovat a předávat, ale i získávat. Pokud bude programování stále školami odsuzováno a odsunováno do pozadí, tato situace se nezlepší. Pokud ale na programování bude nahlíženo jako na něco zábavného a výsledky nebudou vynucovány podmíněným trestem (např. špatnou známkou za nesplněnou činnost na 100 %), programování se stane více zábavnou činností. Lidé, které tato

činnost chytí už v mládí, budou mít lepší možnosti v budoucnosti a nebudou nálepkování jako *„podivíni s mastnými vlasy, bez sociálního života“*.

7.3. ZAČAROVANÝ KRUH – K ZAMYŠLENÍ...

Pokud má mít společnost dobré informatiky/programátory, musí mít takový informatik/programátor dobré vzdělání. Pokud informatik/programátor dostane velice dobrý plat ve firmě, ale ve škole dobrý plat nedostane, nepůjde nikdy dobrý informatik/programátor učit. Pokud nebude nikdy přítomen dobrý informatik/programátor ve škole, aby učil nové nástupce, bude stále nedostatek dobrých informatiků/programátorů, protože nikdo nebude dělat tu *„samou“* práci za míň peněz.

Ačkoliv se pár předešlých řádku tváří, jako *„komáří problém“* tak opak je pravdou. Podobná situace může nastat např. ve zdravotnictví. Pokud budou chybět peníze doktorům, půjdou tito lidé pracovat jinam, příkladem přímo ze života může být situace, která nastala v ČR před pár lety, kdy si stěžující doktoři nevymohli stávkami a protesty vyšší platy. Následoval pak odliv těchto pracovníku do ciziny, většinou do Německa. Vzápětí se zjistilo, že tyto lidi potřebujeme, situace se tak velice rychle vyřešila tím, že se zmiňovaným profesím zvedly platy.

Co ale platy učitelů? Tento problém totiž nezjistí společnost po 2-5 letech, nýbrž po celých dekádách a v tomto případě nebude náprava tohoto problému tak snadná a rychlá, jako v případě výše zmíněných doktorů.

Závěr

Kvalifikační práce *„Programování dříve a dnes, změny přístupu ve výuce programování v ČR“* podala čtenářovi informace ohledně problematiky programování, a to nástin programování a jeho historii, čtenář se dozvěděl o celkovém vývoji, od počátečních lehčích forem programování až do přítomnosti a bylo mu sděleno, jakým způsobem se nejspíše bude programování ubírat v budoucnosti, kdy byla prognóza založena na sledování aktuálních trendů a dění ve světě ICT. V práci jsou zmíněny aktuální přístupy ve výuce programování a v programování samotném, v krátkosti jsou informace doplněny o výhody i nevýhody každého způsobu. Nechybí ani pohled na aktuální situaci na trhu práce v oblasti ICT a programování, doplněný o informace o užití jednotlivých programovacích jazyku, a i možnosti potencionálního výdělku za zhotovenou práci v určitém programovacím prostředí. Programování také bylo komparováno s pouhou schopností digitální technologie obsluhovat – ne programovat. S programováním souvisí i *„online svět“*, kdo je v tomto případě offline, nemá tak velké možnosti, protože programování je silně spjato s internetem a online stavem.

Kvalifikační práce se pak postupně přesouvala od programování obecně k problematice programování v ČR, tedy výzkumu vybraných škol. Výuka ICT je v ČR ukotvena zákonem, kdy je obsažena v rámcovém vzdělávacím programu, který je závazný pro instituce regionálního školství, kterým se tedy řídí základní školy a školy střední. Školy si na základě RVP povinně vytvářely svoje ŠVP a zde můžeme právě činnost programování najít ve vzdělávací oblasti *„informační a komunikační technologie“*. Výzkum nebyl zaměřen na základní školy, ale na školy střední a gymnázia. Byly získávány informace o tom, jak probíhala výuka programování před zavedením RVP, jak výuka probíhá nyní, v době RVP/ŠVP, a jakým způsobem tyto školy chtějí vzdělávací oblast *„informační a komunikační technologie“* zlepšovat a do budoucna aktualizovat vzhledem ke Strategie digitálního vzdělávání do roku 2020. Výsledky výzkumu byly podány slovně i za pomocí kontingenčních tabulek a grafů a v diskuzi krátce sumarizovány a doplněny o vlastní komentář autora.

Vzhledem k tomu, že plán Strategie digitálního rozvoje do roku 2020 ještě není zcela realizován a školy ve většině případů teprve plánují jeho realizaci (jaké změny budou chystat a jak je budou realizovat), naskýtá se možnost na tuto kvalifikační práci navázat v době, kdy již bude strategie pilotně zavedena nebo v pozdější době, kdy již bude plán ukotven stabilně. Na práci lze v tomto případě navázat a komparovat změny, které vznikly v tomto časovém období a reagovat tak přesněji na vývoj programování na školách. Dále se např. ukazuje v možnost mapovat výuku programování již na základních školách (forma programování hrou), protože i tímto směrem se nejspíše výuka ICT ponese.

Seznam zdrojů

- ACM, IEEE Computing Curricula 2001. Association for Computing Machinery.[online] 2001 [cit. 2018-15-01] dostupné online: https://www.acm.org/binaries/content/assets/education/curricula-recommendations/cc2001.pdf
- Assembly language. IBM [online]. USA: IBM, 2014 [cit. 2017-11-25]. Dostupné z: https://www.ibm.com/support/knowledgecenter/SSLTBW_2.1.0/com.ibm.zos.v2r1.asma400/asmr102112.htm
- BACKUS, J. W., J. H. WEGSTEIN, A. VAN WIJNGAARDEN, et al. Report on the algorithmic language ALGOL 60. Communications of the ACM [online]. 3(5), 299-314 [cit. 2018-03-13]. DOI: 10.1145/367236.367262. ISSN 00010782. Dostupné z: http://portal.acm.org/citation.cfm?doid=367236.367262
- BAJTLER, Martin. Školy informatiku neřeší, říkají zakladatelky kurzů programování pro děti. IDnes.cz [online]. Praha: MAFRA, 2017, 22. 10. 2017 [cit. 2018-02-05]. Dostupné z: https://praha.idnes.cz/programovani-informatika-skola-kurz-klara-stouracova-silvie-zemanova-deti-1b4-/praha-zpravy.aspx?c=A171020_359319_praha-zpravy_rsr
- BĚHÁLEK, Marek. Programovací jazyky [Online prezentace]. Ostrava: Katedra informatiky, FEI VŠB, [citováno 4. 11. 2017]. Dostupné online: http://wiki.cs.vsb.cz/images/4/4c/Progjaz.pdf
- Comparing Python to Other Languages. *Https://www.python.org* [online]. Python.org, 2018, nedat. [cit. 2018-03-13]. Dostupné z: https://www.python.org/doc/essays/comparisons/
- Dědičnost a polymorfismus. ITnetwork.cz [online]. ČR: ITnetwork.cz, 2012 [cit. 2017-11-25]. Dostupné z: https://www.itnetwork.cz/csharp/oop/c-sharp-tutorial-dedicnost-a-polymorfismus
- Definition of: compiler. PCMag [online]. New York, United States: Ziff Davis, 2017 [cit. 2017-11-04]. Dostupné z: https://www.pcmag.com/encyclopedia/term/40105/compiler
- DOSTÁL, J. Informační a počítačová gramotnost – klíčové pojmy informační výchovy. In Infotech 2007 – moderní informační a komunikační technologie ve vzdělávání. Olomouc: Votobia, 2007. s. 60–65. ISBN 978-80-7220-301-7.
- DRESLER, Robert. Proč vývojáři odcházejí z IT firem. Robert Dresler: Vývoj software není jenom o kódování… [online]. 2014, 12. 12. 2014 [cit. 2018-03-13]. Dostupné z: http://www.robertdresler.cz/2014/01/proc-vyvojari-odchazeji-z-it-firem.html
- FELDMAN, Brian. Why don't more people work as programmers? Forbes [online]. 499 Washington Blvd Jersey City, NJ: Forbes, 2014 [cit. 2018-01-20]. Dostupné z:

https://www.forbes.com/sites/quora/2014/10/31/why-dont-more-people-work-as-programmers/#2d1b1be055c0

- FERGUSON, Andrew.A History of Computer Programming Languages. [Online] 2000. [Citace: 4. 11. 2017.] http://www.contrib.andrew.cmu.edu/~sfpeer/ProgrammingLanguages.html
- Fortran creator John Backus dies: Programming innovator made computer coding easier, more intuitive. NBC News [online]. New York, USA: NBC News, 2017 [cit. 2017-11-04]. Dostupné z: http://www.nbcnews.com/id/17704662/
- FREEDMAN, Jeri. Careers in Computer Science and Programming. New York: The Rosen Publishing Group, 2011. ISBN 978-1-4488-1318-6.
- GEORGE A. FIERHELLER. Do not fold, spindle or mutilate: the "hole" story of punched cards. Markham, Ont: Stewart Pub, 2006. ISBN 18-941-8386-X.
- GLEICK, James. The information: a history, a theory, a flood. New York: Vintage Books, c2011. ISBN 978-1400096237.
- High level languages vs Low level languages. EDUCBA [online]. EDUCBA, 2015 [cit. 2017-11-04]. Dostupné z: https://www.educba.com/high-level-languages-vs-low-level-languages/
- Nedostatkovým zbožím v tuzemsku jsou programátoři, nároky a platy rostou. Hospodářské Noviny: ICT Revue [online]. Praha: Economia, 2018, 16. 6. 2015 [cit. 2018-03-13]. Dostupné z: https://archiv.ihned.cz/c1-64175890-nedostatkovym-zbozim-v-tuzemsku-jsou-programatori-naroky-a-platy-rostou
- ICT ve vzdělávání. *MŠMT* [online]. Praha: MŠMT, 2018, 2009 [cit. 2018-03-13]. Dostupné z: http://www.msmt.cz/ict
- IT profese. Zkus IT: rozhodni se pro budoucnost [online]. Praha: Zkus IT o.s., 2010, nedat. [cit. 2018-03-13]. Dostupné z: http://www.zkusit.cz/profese/sw.php
- JANOVSKÝ, Dušan. Úvod do JavaScriptu. Jak psát web: Jak psát web o tvorbě, údržbě a zlepšování internetových stránek [online]. Praha, 2014, nedat. [cit. 2018-03-13]. Dostupné z: https://www.jakpsatweb.cz/javascript/javascript-uvod.html
- KLADIVOVÁ, Barbora. České firmy mají nedostatek IT specialistů, podle odhadů chybí až 100 tisíc programátorů. IRozhlas.cz [online]. Praha: Český rozhlas, 2018, 10. 5. 2016 [cit. 2018-03-13]. Dostupné z: https://www.irozhlas.cz/zpravy-domov/ceske-firmy-maji-nedostatek-it-specialistu-podle-odhadu-chybi-az-100-tisic-programatoru-_201604100315_amanourova
- Kteří programátoři berou nejvíc? ITtalents: > $ getent group it-specialists [online]. Radlická 663/28, Praha 5: Recruiting Talents, 2017 [cit. 2018-01-31]. Dostupné z: https://www.ittalents.cz/cz/blog/23-kteri-programatori-berou-nejvic/

- Microsoft is deleting its AI chatbot's incredibly racist tweets. Business Insider [online]. New York: Business Insider, 2009, 24. 3. 2016 [cit. 2018-01-20]. Dostupné z: http://uk.businessinsider.com/microsoft-deletes-racist-genocidal-tweets-from-ai-chatbot-tay-2016-3
- MOORE, Gordon. Cramming more components onto integrated circuits. Electronics [online]. 1965, 38(8), 4 [cit. 2018-01-20]. Dostupné z: https://drive.google.com/file/d/0By83v5TWkGjvQkpBcXJKT1I1TTA/view
- NAVRÁTIL, Jiří. Kdy přijdou programátoři o práci? Lupa.cz: Server o českém internetu [online]. Milady Horákové 116/109 160 00 Praha 6: Internet Info, 2018, 1. 7. 2003 [cit. 2018-01-25]. Dostupné z: https://www.lupa.cz/clanky/kdy-prijdou-programatori-o-praci/
- NORRIS, Pippa. Digital divide: civic engagement, information poverty, and the Internet. Reprint. Cambridge: Cambridge University Press, 2001. ISBN 0521002230.
- PARKER, Matt. Things to make and do in the fourth dimension: a mathematician's journey through narcissistic numbers, optimal dating algorithms, at least two kinds of infinity, and more. New York: Farrar, Straus and Giroux, 2014. ISBN 03-742-7565-3.
- PATTIS, Richard E., Jim ROBERTS a Mark. STEHLIK. Karel the robot: a gentle introduction to the art of programming. 2nd ed. /. New York: Wiley, c1995. ISBN 0471597252.
- PAULENKOVÁ, Kristína. Výuka informatiky na školách se mění, zaměří se na programování. Ihned.cz [online]. Praha: Economia, 2018, 11. 12. 2017 [cit. 2018-02-06]. Dostupné z: https://archiv.ihned.cz/c1-65984410-vyuka-informatiky-na-skolach-se-meni-zameri-se-na-programovani
- Průměrné mzdy 3. čtvrtletí 2017. Český statistický úřad [online]. Praha: Český statistický úřad, 2017, 4. 12. 2017 [cit. 2018-03-13]. Dostupné z: https://www.czso.cz/csu/czso/cri/prumerne-mzdy-3-ctvrtleti-2017
- Rámcový vzdělávací program pro základní vzdělávání. [online]. Praha: MŠMT, 2017. 162 s. [cit. 2018-02-05]. Dostupné online: http://www.nuv.cz/uploads/RVP_ZV_2017.pdf
- SHURKIN, Joel N. *Engines of the mind: the evolution of the computer from mainframes to microprocessors*. [Updated pbk. ed.]. New York: Norton, c1996. ISBN 0-393-31471-5.
- STRATEGIE DIGITÁLNÍHO VZDĚLÁVÁNÍ DO ROKU 2020. [online]. Praha: MŠMT, 2014. 49 s. [cit. 2018-02-06]. Dostupné online: http://www.vzdelavani2020.cz/images_obsah/dokumenty/strategie/digistrategie.pdf
- TANZ, Jason. Soon We Won't Program Computers. We'll Train Them Like Dogs. Wired [online]. New York: Wired, 2018, 17. 5. 2016 [cit. 2018-03-13]. Dostupné z: https://www.wired.com/2016/05/the-end-of-code/

- The Ideal HPC Programming Language: Maybe it's Fortran. Or maybe it just doesn't matter. Acmqueue [online]. acmqueue, 2010 [cit. 2017-11-04]. Dostupné z: http://queue.acm.org/detail.cfm?id=1820518
- The Modern History of Computing. Stanford Encyclopedia of Philosophy [online]. Stanford: Center for the Study of Language and Information (CSLI), Stanford University, 2006 [cit. 2017-10-13]. Dostupné z: https://plato.stanford.edu/entries/computing-history/
- The State of the Octoverse 2017. GitHub Inc. [online]. San Francisco, Kalifornie, USA: GitHub, 2018, 2017 [cit. 2018-02-15]. Dostupné z: https://octoverse.github.com/
- TIŠNOVSKÝ, Pavel. Programovací jazyky z vývojářského pekla. Root.cz: Informace nejen ze světa Linuxu [online]. root.cz, 2018, 28. 4. 2016 [cit. 2018-01-30]. Dostupné z: https://www.root.cz/clanky/programovaci-jazyky-z-vyvojarskeho-pekla/
- Umělá inteligence od Google učí své potomky. Programuje jiné umělé inteligence lépe než lidé. Česká televize [online]. Praha: Česká televize, 20. 10. 2017 [cit. 2018-01-20]. Dostupné z: http://www.ceskatelevize.cz/ct24/veda/2279656-umela-inteligence-od-google-uci-sve-potomky-programuje-jine-umele-inteligence-lepe-nez
- Úvod do objektově orientovaného programování v C#. ITnetwork.cz [online]. ČR: ITnetwork.cz, 2012 B [cit. 2017-11-25]. Dostupné z: https://www.itnetwork.cz/csharp/oop/c-sharp-tutorial-uvod-do-objektove-orientovaneho-programovani
- VERMEULEN, Alessandro. Why you should switch to declarative programming. Alessandro Vermeulen: Functional Programming and Reactive Systems Evangelist [online]. Alessandro Vermeulen, 2018, 19. 5. 2013 [cit. 2018-01-30]. Dostupné z: https://alessandrovermeulen.me/2013/05/19/why-you-should-switch-to-declarative-programming/
- WEIK. Ballistic Research Laboratories Report No. 971: a Survey of Domestic Electronic Digital Computing Systems. Aberdeen Proving Ground [online]. United States Department of Commerce Office of Technical Services, 1955, 41 [cit. 2017-11-04]. Dostupné z: http://ed-thelen.org/comp-hist/BRL-e-h.html#ENIAC
- WEXELBLAT, Richard L. History of programming languages. New York: Academic Press, 1981. ACM monograph series. ISBN 01-274-5040-8.
- What is Coding? Yearofcodes [online]. Birmingham: yearofcodes, 2014 [cit. 2017-10-13]. Dostupné z: http://yearofcodes.tumblr.com/what-is-coding
- WILSON, L. B. ALGORITHMS, Robert Sedgewick, Addison-Wesley Publishing Company, 1983. No. of pages: 552. Price. Software: Practice and Experience [online]. 1984, 14(5), 501-502 [cit. 2017-10-13]. DOI: 10.1002/spe.4380140510. ISSN 00380644. Dostupné z: http://doi.wiley.com/10.1002/spe.4380140510

Seznam obrazových materiálů, tabulek a grafů

Seznam příloh

PŘÍLOHA Č. 1: OTÁZKY PRO INTERVIEW A EMAILOVÁ FORMA DOTAZNÍKU

Po domluvení schůzky s učitelem jsem ho navštívil v jeho volném čase a pokládal otázky uvedené níže. Průběh nebyl striktní a nemusel probíhat vždy stejně, ale zodpovězeny byly všechny body, někdy byly získány i informace navíc. Dotazník jsem se snažil vytvořit tak, aby jeho zodpovídání nebylo náročné, ani zdlouhavé a pedagogy tak nezdržoval víc, než by bylo třeba.

1) Jméno školy a adresa.
    - Zaměření školy (obecné, jazykové, technické).
    - Studované obory a jejich délka.
    - Časová dotace ICT pro jednotlivé studijní obory.
2) Probíhala u Vás výuka programování před zavedením RVP a ŠVP?
    - Jaký programovací jazyk byl pro výuku použit?
3) Probíhá u Vás výuky programování nyní?
    - Jaký programovací jazyk je užit pro výuku programování?
    - Probíhá v rámci předmětu ICT nebo jiného předmětu?
    - V jakém ročníku se žáci k samotnému programování dostanou?
    - Rozdíly v přístupu k programování vzhledem k různým oborům?
    - Vede Vaše škola „seminář ICT“? Pokud ano, jaká je jeho dotace?
    - Maturitní zkouška z programování? Jakým způsobem se uskutečňuje?
4) Strategie digitálního vzdělávání do roku 2020.
    - Víte, o čem je řeč? (pokud ne, bylo vysvětleno).
    - Chystá Vaše škola nějaké změny ve výuce ICT? (časová dotace, změna způsobu výuky, přerozdělení učiva do průřezových témat apod.).
    - Chystá Vaše škola změny ve výuce programování?
5) Prostor pro učitele ICT a jeho poznámky a jiné záležitosti, které nešly přímo zařadit jinam, nebo se na ně narazilo v průběhu rozhovoru.

Emailová forma dotazníku zněla podobně, až na to, že obsahovala i úvodní slovo pro sekretariát školy, protože v některých případech škola neposkytovala přímo email na učitele, nebo webové stránky neobsahovaly informace, který učitel je učitelem ICT. V úvodním slovu jsem se představil jako student UP, podal informace o mé osobě a mém výzkumu k diplomové práci a celkovém záměru kvalifikační práce. Následoval dotazník s otázkami, po nich závěr s poděkováním. V některých případech probíhala ještě následná korespondence, kdy se učitelé více zajímali a chtěli vědět některé informace navíc.

ANOTACE

Jméno a příjmení:	Ladislav Orel
Katedra:	Katedra technické a informační výchovy
Vedoucí práce:	doc. PhDr. Milan Klement, Ph.D
Rok obhajoby:	2018

Název práce:	Programování dříve a dnes, změny přístupu ve výuce programování v ČR
Název práce v anglickém jazyce:	Programming earlier and today, changes in attitudes in the teaching of programming in the Czech Republic.
Anotace práce:	Kvalifikační práce je zaměřena na činnost programování a jeho postupný vývoj v čase, přičemž jsou zmíněny nejen programovací jazyky, ale i způsoby programování. Předmětem výzkumné části je zhodnocení aktuálního stavu výuky programování v ČR na středních školách.
Klíčová slova:	Programování, historie programování, programování v ČR, paradigmata programování, informační a komunikační technologie, ICT, programování na středních školách, výzkum zaměřený na programování středních škol.
Anotace v anglickém jazyce:	Master thesis is focused on programming and its evolving in time. Moreover there are descripteds many types of programming paradigmas. Research is focused on high schools and their ways of teaching programming in ICT subjects.
Klíčová slova v anglickém jazyce:	Programming, history of programming, programming in Czech republic, paradigm of programming, information and communication technologies, ICT, programming at secondary schools, research about programming manners at high schools in Czech republic.
Přílohy vázané v práci:	1 – dotazník pro zisk informací
Rozsah práce:	64 stran
Jazyk práce:	Český