TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta přírodovědně-humanitní
a pedagogická

Programování v pregraduální přípravě učitelů informatiky[verze 16. 5.]

Diplomová práce

Studijní program: N1101 – Matematika
Studijní obory: 7504T077 – Učitelství informatiky pro střední školy2
7504T089 – Učitelství matematiky pro střední školy

Autor práce: **Bc. Ondřej Vraštil**
Vedoucí práce: Mgr. Jan Berki, Ph.D.

Liberec 2016

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Science, Humanities
and Education

Programming in undergraduate training of CS teachers

Master thesis

Study programme: N1101 – Mathematics
Study branches: 7504T077 – Teacher training for lower and upper-secondary school, Informatics
7504T089 – Teacher Training for Upper Secondary Schools, Mathematics

Author: **Bc. Ondřej Vraštil**
Supervisor: Mgr. Jan Berki, Ph.D.

Liberec 2016

Tento list nahraďte
originálem zadání.

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum:

Podpis:

Abstrakt

Tato zpráva popisuje třídu `tulthesis` pro sazbu absolventských prací Technické univerzity v Liberci pomocí typografického systému LaTeX.

Abstract

This report describes the `tulthesis` package for Technical university of Liberec thesis typesetting using the LaTeX typographic system.

Poděkování

Rád bych poděkoval všem, kteří přispěli ke vzniku tohoto dílka.

Obsah

1 Úvod **8**

2 Programování **10**
2.1 Algoritmus a algoritmizace – definice pojmů 10
2.2 Programování a jeho vztah k algoritmizaci 11
2.3 Programovací paradigmata 13
2.4 Programování z pohledu deklarovaného kurikula ZŠ a SŠ 14
2.4.1 Deklarované kurikulum výuky informatiky na Slovensku . . . 17
2.5 Proč učit programování? 22
2.6 Jak učit programování? 23

3 Analýza deklarovaného kurikula pregraduální přípravy **27**
3.1 Stav problematiky 27
3.2 Pregradualní příprava učitelů informatiky – popis zkoumaného vzorku 27

4 Textová analýza **32**
4.1 Popis vzorku 32
4.2 Popis způsobu analýzy 32
4.3 Výsledky 32

5 Návrh konceptu pregraduální přípravy **33**
5.1 ZŠ 33
5.2 SŠ 33

6 Závěr **34**

1 Úvod

Nedílnou součástí pregraduální přípravy budoucích učitelů informatiky základních i středních škol je výuka programováni a teorie algoritmů. Toto téma je v současnosti získává na popularitě i na středních a základních školách, kde je mu věnováno stále více pozornosti. Výuka úvodu do programování a algoritmického myšlení je nejen obsažena v rámcovém vzdělávacím programu pro gymnázia a některé odborné střední školy, ale v uzpůsobené podobě se objevuje i ve výuce na školách základních, kde se mohou mladí žáci setkat se speciálními programovacími jazyky pro děti.

Aby mohla být správně provedena didaktická transformace směrem od učiva k žákovi, musí její původce – učitel mít dostatečný vhled do problematiky a patřičné vzdělání v příslušném tématu. Jednou z hlavních premis je kvalitní vzdělání v rámci pregraduální přípravy, ve které by měl student – budoucí učitel získat vhled do tématu natolik, aby dokázal obstojně předat znalosti svým žákům. Na kvalitu pregraduální přípravy má vliv mnoho aspektu, jedním z nich je i relevance a aktualita1.čeho ale? upravit, které mají pro informatiku, jakožto mladý a dynamicky se rozvíjející obor velký význam. Fakulty připravující učitele informatiky by měly pružně reagovat na moderní trendy ve výuce a uspokojit poptávku studentů po kvalitním a hlavně využitelném vzdělání, které budou moci využít ve své budoucí praxi,nehledě na to, že z některých studentů se mohou stát i vysokoškolští pedagogové. V dnešní době se na našich základních a středních školách začínají využívat pro školní prostředí nová témata jako je robotika nebo unplugged teaching, pro která je znalost algoritmizace nutná. 2.trošku věta navíc

Jak si naše vysoké školy vedou ve výuce programování a algoritmizace? Následují moderní trendy a požadavky zaměstnavatelů budoucích absolventů? Je pořadí předmětů během studia smysluplné? Dá se vysledovat podobnost mezi programy napříč republikou? Existuje jeden nejvhodnější způsob jak učit programovaní budoucí učitele, nebo je možnost volit z více cest?

Abychom tyto otázky mohli zodpovědět, je v první řadě nutné pokusit se analyzovat zdroje týkající se programování a pregraduální přípravy učitelů. Dále je potřeba získat data o obsahu jednotlivých předmětů v rámci pregraduální přípravy napříč fakultami v ČR. Fakulty tyto data zveřejňují v sylabech, které jsou volně k dispozici na stránkách jednotlivých fakult. Získaná data budou zkoumána formou textové analýzy, jež by mohla na výše zmíněné otázky odpovědět. Na výsledcích teoreticko-rešeršní části i praktické části bude postavena závěrečná část, návrh ideálního sestavení vhodného konceptu výuky programování na VŠ pro budoucí učitele středních i základních škol.

ToDo

P.

1. čeho ale? upravit . . . 8
2. trošku věta navíc . . . 8
3. opakuje se na konci odstavce . . . 10
4. je to takto ok? malá písmena na začátku, bez teček a čárek . . . 11
5. ???tahle kapitola by potřebovala trochu vyladit a zpřehlednit, . . . 14
6. upravit- zavazné není učivo ale výstupy - nutné přepracovat . . . 14
7. lépe . . . 15
8. citace . . . 15
9. citace . . . 18
10. tabulka pro tabulku, bud využít nebo smazat či zkrátit . . . 18
11. zkusit zahrnout i Brity . . . 21
12. Chybí citace, je potřeba trošku uhladit . . . 22
13. citace - stránky blueJ . . . 23
14. Dopsat že kvůli tomu by určitě měla být součástí didaktika informatiky . . . 26
15. mozno se ještě dale oprit o clanek Kalase o didkatice programování z Didinfa 2009 . . . 26
16. dát oficální názvy nebo nechat názvy s městy????? . . . 28
17. Srovnat s kapitolou 3.1 . . . 32
18. Ještě tři podkapitoly:1Jaká je dotace předmětů Jaká je jejich struktura (jak na sebe navazují, jak jsou oddělené, jestli jsou algoritmy zvlašt nebo dohromady s programováním) Obsah předmětů . . . 32

2 Programování

Pokud máme zkoumat pregraduální přípravu učitelů informatiky v oblasti programování, měli bychom odpovědět na důležitou otázku – Proč se vlastně zařazuje výuka programování do přípravy budoucích učitelů informatiky? Jaký zde má smysl?**3.opakuje se na konci odstavce** Definujeme nejdřív co je algoritmizace a programování, jak jsou tato témata zakotvena v našich kurikulárních dokumentech a nakonec odpovíme na dvě důležité otázky: Proč by vlastně mělo být součástí pregraduální přípravy budoucích učitelů informatiky na VŠ a jakým způsobem může být vyučováno.

2.1 Algoritmus a algoritmizace – definice pojmů

> *„Before there were computers, there were algorithms. But now that there are computers, there are even more algorithms, and algorithms lie at the heart of computing. “*
>
> *- Introduction to Algorithms*

Algoritmus je jeden z ústředních (ne li ten vůbec nejústřednější) pojem z oblasti informatiky (Schubert & Schwill 2011). V literatuře najdeme několik jeho definic. Schubert a Schwill (2011, s. 4) popisují algoritmus jako formálními prostředky popsatelný, mechanicky proveditelný postup k řešení třídy problému. Cormen (2013, s. 1) popisuje algoritmus jako sadu kroků ke splnění úlohy, počítačový algoritmus jako sadu korků ke splnění úlohy tak přesně popsaných, aby je dokázal vykonat počítač. Pro Skienu (2008, s. 3) je algoritmus procedura ke splnění konkrétní úlohy a myšlenka, která stojí za počítačovým programem. Cormen, Leiserson, Rivest[1] a Stein (2009, s. 5) ve své obsáhlé publikaci popisují algoritmus jako jasně definovanou výpočetní proceduru, která přijímá hodnotu nebo soubor hodnot jako vstup a produkuje hodnotu nebo soubor hodnot jako výstup. Známý informatik Donald Knuth (2008, s. 5), tvůrce typografického systému Tex[2] definuje algoritmus jako konečnou množinou pravidel, která popisují posloupnost operací pro řešení jistého typu problémů. Další definici najdeme u Harela a Feldmana (2004, s. XII): Algoritmus je abstraktní návod předepisující proces, který by mohl být proveden člověkem, počítačem nebo jinými prostředky. Podle Knutha (2008, s. 4-6) musí algoritmus splňovat několik základní vlastností:

[1] písmeno „R“ v RSA
[2] kterým je psána i tato práce

- *konečnost* – algoritmus musí vždy po určitém počtu kroků skončit
- *určitost* – každý krok algoritmu musí být přesně definován a pro každý případ v něm musí být s určitostí a jednoznačností popsány prováděné operace
- *vstup* – každý algoritmus má nula nebo více vstupů: to jsou veličiny, které do algoritmu zadáme před jeho zahájením nebo které načteme dynamicky za běhu
- *výstup* – algoritmus má také jeden nebo více výstupů: to jsou veličiny, které mají zadaný vztah ke vstupům
- *efektivita* – algoritmus by měl být zároveň efektivním, což znamená, že všechny jeho operace musí být v rozumné míře jednoduché, takže by je v principu měl být schopen přesně a za konečnou dobu provést kdokoli s tužkou a papírem

4.je to takto ok? malá písmena na začátku, bez teček a čárek

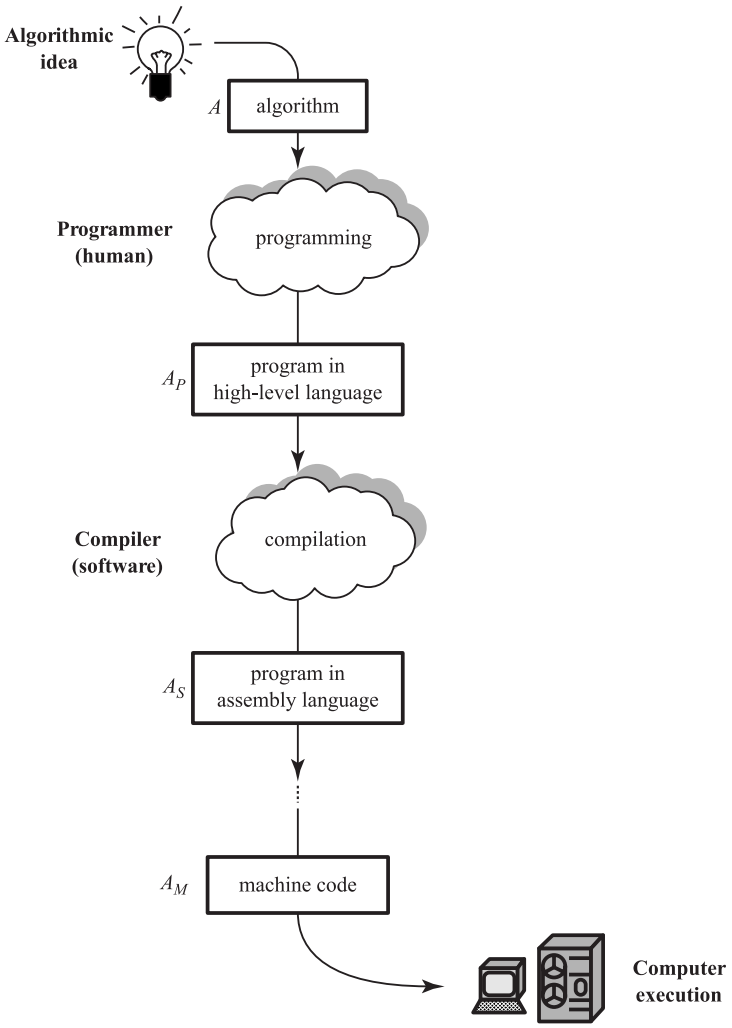
Často najdeme přirovnání algoritmu ke kuchařskému receptu jako například u Knutha (2008, s. 6) nebo u Harela a Feldmana (2004, s. 4). Vstupem jsou pak v tomto případě suroviny, výstupem je hotové jídlo. Recept je konečný, naše vaření nebude probíhat nekonečně dlouho a efektivní – můžeme je provést v relativně krátkém čase.

Proces převodu problému na jednotlivé kroky nazýváme **algoritmizace** (Schubert & Schwill 2011, s. 67). Motyčka (1999, s. 5) chápe algoritmizaci problému při tvorbě programu jako vyvážení postupu řešení daného problému na počítači, dále jako *krásnou tvůrčí činnost, při které využíváme intelekt, zkušenosti, intuici a postupně vytváříme spoustu postupů řešení, z nichž ty počáteční zpravidla k cíli nevedou vůbec, další vedou k cíli pouze občas a havárie již jen v určitých zvláštních (mezních) situacích, o kterých jsme na počátku našeho snažení vůbec neuvažovali.*

Jak je z výše uvedených definic patrné, pojmy algoritmus a algoritmizace jsou důležitou součástí informatiky, ale můžeme říci, že ji i přesahují, protože algoritmem můžeme popsat například proces vaření. Pro potřeby této práce je potřeba správně rozhodnout, které předměty v průběhu pregraduální přípravy pomáhají studentovi v rozvíjení jeho schopnosti algoritmizace problémů, protože to v sylabech těchto předmětů nemusí být explicitně zmíněno – stejně tak, jako když se při výuce vaření na ZŠ o algoritmizaci nezmiňujeme, ačkoliv se o algoritmizaci jedná.

2.2 Programování a jeho vztah k algoritmizaci

Pokud jsme si definovali algoritmus obecně, musíme definovat i propojení tohoto pojmu s počítači – stroji. Počítače v současné době slouží k řešení mnoha problémů okolo nás, např. automatickému pilotování letadla, nebo složitým simulacím chemických procesů. (Harel & Feldman 2004, s. 49). Vše jsou složité algoritmy vykonávané počítačem. Aby počítač mohl provést příkaz, je algoritmus zapsán v **programovacím jazyku**, umělém jazyku, který je zpravidla zredukovanou podmnožinou anglického jazyka. Programem pak rozumíme zápis algoritmu v programovacím jazyku (Motyčka 1999, s. 6). V současné době existuje mnoho

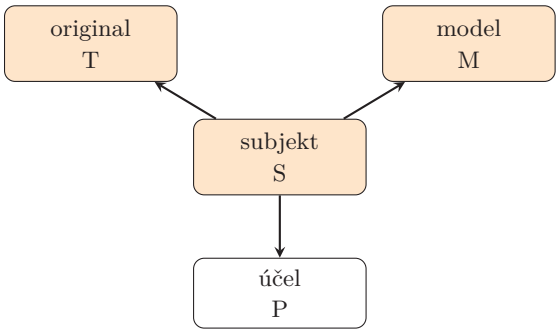


Obrázek 2.1: Proces provedení algoritmu počítačem (Harel & Feldman 2004, s. 56)

jazyků lišících se syntaxí a použitím. Jazyky se liší mírou abstrakce – vyšší jazyky se podobají lidské řečí, jsou více abstraktní, naopak pomocí nižších jazyků je možné lépe ovládat strojové operace, mají nižší abstrakci. Schéma (2.1) popisuje celý princip od myšlenky algoritmu až po jeho provedení počítačem. Programátor – člověk naprogramuje algoritmus do programu v některém z vyšších jazyků. Poté si počítač převádí, tzv. kompiluje tento program do nižšího jazyku, tzv. jazyku symbolických adres (anglicky assembly language). Tento jazyk už přímo odkazuje na místa v počítačové paměti. Následně už je program převeden do strojového kódu, jehož instrukcím rozumí procesor a je schopen je provést. Alternativa ke kompilaci je převod pomocí interpretru. Na rozdíl od kompilace dokáže interpretr provést program aniž by byl převeden do strojového kódu (Harel & Feldman 2004, s. 54-57)

2.3 Programovací paradigmata

V jistém smyslu můžeme na informatiku nahlížet jako na vědu, která se zabývá vytváření realizovatelných modelů. Proces vytváření modelů se dá popsat jako relace R(S,P,T,M), kde subjekt S vytváří model M podle originálu T za účelem P (Schubert & Schwill 2011, s. 135). Jelikož pomocí počítače modelujeme mnoho různých situací, existuje mnoho programovacích jazyků. Ty se sdružují do několika hlavních stylů, které se liší přístupem k abstrakci dat a operacím s nimi a představuji specifický přístup k programu a jeho provedení. Takovéto styly nazýváme programovací paradigmata (Bolshakova 2005, s. 286).



Obrázek 2.2: Schéma vytváření modelů (Schubert & Schwill 2011, s. 135)

Imperativní paradigma se vyvinulo z nižších jazyků – strojového a jazyku symbolických adres (Bolshakova 2005, s. 286). Z pohledu toho paradigmatu je počítač soubor paměťových buněk organizovaných do různých typů datových struktur, jako pole nebo seznam. Program v tomto paradigmatu je sekvence přesných příkazů, která mění data v těchto strukturách a které jsou spuštěny v přesném sledu. Jelikož program pracuje se stále stejným paměťovým místem, ke kterému uživatel přistupuje pomocí tzv. proměnných, měl by mít programátor v každém kroku programu přehled, jak se tyto hodnoty mění.

Ve **Funkcionálním paradigmatu** je programem výraz, který se skládá z funkcí s podobnou strukturou jako mají funkce matematické. Jedna funkce může být argumentem jiné funkce. Po spuštění proběhne zjednodušení tohoto výrazu až do podoby, kdy zjednodušit dále nelze. Neexistuje zde princip modifikovatelné paměti jako v imperativním paradigmatu (Škavrda 2006).

V **logickém paradigmatu** je program brán jako soubor logických formulí – axiomů, které popisují vztahy a vlastnosti nějakých objektů a dotaz, který má být na základě těchto pravidel zodpovězen – dokázán. Základní myšlenka **objektově orientovaného paradigmatu** se zakládá na představě, že reálný svět je množství objektů, které spolu navzájem komunikují. Definujeme zde skupiny prvků s podobnými vlastnostmi nazýváme třídy. Na rozdíl od imperativního paradigmatu, ve kterém jsou funkce považovány za aktivního činitele modifikujícího pasivně uložená data, objektově orientované paradigma považuje za aktivní v paměti uložené objekty a za funkce jsou jen pasivní zprávy, které si objekty mezi sebou posílají. Přehlednou tabulku paradigmat uvádí Bolshakova (2005, s. 287):

Paradigma	Klíčový koncept	Program	Provedení programu	Výsledek
Imperativní	Příkaz (instrukce)	Sekvence příkazů	Provedení příkazů	Zavěrečný stav počítačové paměti
Funkcionální	Funkce	Soubor funkcí	Vyhodnocení funkcí	Hodnota hlavní funkce
Logické	Predikát	Logické formule: axiomy a theorémy	Logické dokazování theorému	Úspěch nebo neúspěch dokazování
Objektově orientované	Objekt	Soubor tříd a objektů	Výměna zpráv mezi objekty	Finální stav objektových stavů

Pro úplnost uveďme ještě tabulku, s přehledem programovacích jazyků pro příslušná paradigmata. Tato tabulka samozřejmě není úplná, pro každé paradigma najdeme nespočet jazyků, uvádí pouze základní přehled. V tabulce jsou uvedeny i jazyky, ve kterých můžeme programovat v několika různých paradigmatech, takové jazyky nazýváme **multiparadigmatické**.

Paradigma	Klíčový koncept
Imperativní	C, C++, Java, PHP, Python, Ruby
Funkcionální	Python, Lisp, Scheme
Logické	Prolog
Objektově orientované	C++, Java, Python

Je součástí pregraduální přípravy učitelů v ČR výuka všech paradigmat, nebo se studium zaměřuje pouze na některá? S kterým paradigmatem se seznámí studenti nejdříve? Jsou pro jednotlivá paradigmata vyhrazeny samostatné předměty, nebo je koncentrovaná výuka více paradigmat do jednoho předmětu? To jsou otázky, které můžeme ve výzkumné části zodpovědět.

2.4 Programování z pohledu deklarovaného kurikula ZŠ a SŠ

5.???tahle kapitola by potřebovala trochu vyladit a zpřehlednit,6.upravit- zavazné není učivo ale výstupy - nutné přepracovat Jedním z určujících dokumentů pro vzdělávání na základních a středních školách je Rámcový vzdělávací program. Tento koncepční dokument určuje a specifikuje obsah výuky na republikové úrovni a odvíjí se od něho dokumenty na úrovni nižší – školní vzdělávací programy, má tedy zásadní vliv na podobu výuky informatiky v republice. V jaké míře je zde programování nebo algoritmizace obsažena? Analyzujme nyní tento dokument, abychom zjistili, zda a jak jsou v něm témata algoritmizace a programování obsažena.

V RVP pro základní vzdělávání (MŠMT 2016) spadá pod tzv. vzdělávací oblast Informační a komunikační *Informační a komunikační technologie*. Charakteristika oblasti pojem algoritmus, programování ani jím příbuzná nebo odvozená slova neuvádí. Zaměřuje se spíše na výuku práce s výpočetní technikou a práci s informací. Zmínku najdeme tzv. cílových zaměřeních kdy je uvedeno, že *Vzdělání v oblasti vede žáka k schopnosti formulovat svůj požadavek a využívat při interakci s počítačem algoritmické myšlení.* RVP dále kategorizuje vzdělávací obsah a rozděluje ho do učiva prvního a druhého stupně, pojem algoritmizace ani programování zde nenajdeme.

V RVP pro gymnázia se mění (rozšiřuje) název vzdělávací oblasti na *Informatika a informační a komunikační technologie*. Jak název napovídá, součástí výuky na gymnáziích by měl být základy informatiky jako vědy. Zaměřuje se hlavně na způsob myšlení *Cílem je zpřístupnit žákům základní pojmy a metody informatiky, napomáhat rozvoji abstraktního, systémového myšlení, podporovat schopnost vhodně vyjadřovat své myšlenky, smysluplnou argumentací je obhajovat a tvůrčím způsobem přistupovat k řešení problémů.* Pojem algoritmus se tu objevuje už explicitně také *Žák se seznámí se základními principy fungování prostředků ICT a soustředí se na pochopení podstaty a průběhu informačních procesů, algoritmického přístupu k řešení úloh a významu informačních systémů ve společnosti.* I mezi cíli je algoritmizace zastoupena a to přímo jako bod *uplatňování algoritmického způsobu myšlení při řešení problémových úloh.* Za cíl, který by se k algoritmizaci a programování mohl vztahovat také je *porozumění základním pojmům a metodám informatiky jako vědního oboru a k jeho uplatnění v ostatních vědních oborech a profesích*, tento bod můžeme reprezentovat mnoha způsoby. Ve vzdělávacím obsahu se pak objevuje tématický okruh *Zpracování a prezentace informací*, kde jedním z očekávaných výstupu je, že žák *aplikuje algoritmický přístup k řešení problémů.* To reflektuje i učivo (které je pro tvorbu ŠVP závazné) a jeho bod *algoritmizace úloh – algoritmus, zápis algoritmu, úvod do programování*, na gymnáziu je tedy výuka programování a algoritmizace **povinnou součástí**, ale bližší cíle vzdělávání nejsou popsány.7.lépe

Vzdělávání na odborných středních školách probíhá samozřejmě podle RVP také, každý obor má svůj vlastní dokument. Analyzujme nyní výskyt algoritmizace a programování na RVP oboru Informační technologie 8.citace, kde by úroveň informatického vzdělávání měla být nejvyšší z celého středoškolského systému. Programování najdeme v tzv. klíčových odborných kompetencích, konkrétně klíčová kompetence Programovat a vyvíjet uživatelská, databázová a webová řešení, tzn. aby absolventi:

- algoritmizovali úlohy a tvořili aplikace v některém vývojovém prostředí;
- realizovali databázová řešení;
- tvořili webové stránky.

V RVP pro odborné vzdělávání existují vzdělávací oblasti tak jako v RVP pro gymnázia, dělí se ještě dále na tzv. vzdělávací okruhy podle kterých se na školní úrovni definuje obsah jednotlivých předmětů. Pro programování má odborné vzdělávání samostatný okruh nazvaný *programování a vývoj aplikací* jehož cílem je *naučit žáka vytvářet algoritmy a pomocí programovacího jazyka zapsat zdrojový kód*

programu. V tabulce (2.1) obsažené v RVP najdeme definované výsledky vzdělávání a učivo.

Tabulka 2.1: Obsahový okruh *programování a vývoj aplikací*

Výsledky vzdělávání	Učivo
Žák: • zná vlastnosti algoritmu; • zanalyzuje úlohu a algoritmizuje ji; • zapíše algoritmus vhodným způsobem;	**1 Algoritmizace** • význam, prvky algoritmu
• použije základní datové typy; • použije řídící struktury programu; • vytvoří jednoduché strukturované programy;	**2 Strukturované programování** • datové typy • řídicí struktury
• rozumí pojmům třída, objekt a zná jejich základní vlastnosti; • použije jednoduché objekty;	**3 Úvod do objektového programování** • třída, objekt, vlastnosti tříd
• zná výhody použití jazyka SQL; • použije základní příkazy jazyka SQL;	**4 Základy jazyka SQL** • základní příkazy (SELECT, UPDATE, INSERT, DELETE)
• aplikuje zásady tvorby WWW stránek; • orientuje se ve struktuře HTML stránky; • vytvoří webové stránky včetně optimalizace a validace; • použije formuláře a skriptovací jazyk.	**5 Tvorba statických a dynamických webových stránek**

Pro odborné školy je tedy vzdělávací obsah blíže specifokován. Najdeme zde i zmínky o jednotlivých programovacích paradigmatech.

Shrňme si, jak je programování a algoritmizace obsažena v RVP všech stupňů vzdělávání a jaký to má důsledek na výuku. Celá koncepce RVP dává škole pouze obecný rámec, který by měla dodržovat. Pro jednotlivá témata zde není uvedena časová dotace, což dává možnost upřednostnit některá témata před ostatními. Dále zde nejsou uvedeny metody, které mají být při výuce použity, takže např. algoritmizace může být procvičována mnoha různými způsoby. Pro programování nejsou na národní úrovni určeny ani doporučeny konkrétní programovací jazyky, kromě odborného vzdělávání není ani určeno, jaké paradigma by mělo být při výuce použito. Mezi základními školami vznikají někdy velké rozdíly, kdy některé ŠVP zahrnují programování (skrze dětské programovací jazyky) na prvním stupni, některé vůbec programování nenasazují.[3] Obecně lze říci, že výuka ICT je směřována spíše k ovládání informačních a komunikačních technologií, než k práci s algoritmizací a programováním.

Pokud bychom měli tvořit pregraduální přípravu jen a pouze podle očekávaných výstupů v RVP, v programu pro ZŠ by se výuka programování nemusela

[3]Měl jsem osobní zkušenost na dvou základních školách, jedna s využitím disponibilních hodin vyučovala informatiku během 3.-7. ročníku, kdy už ve třetím ročníku výuka obsahovala dětský programovací jazyk Baltík. Druhá škola měla nejnižší možnou dotaci jednu hodinu pro každý stupeň a o výuce programování se zde vůbec neuvažovalo, jednalo spíše o výuku informačních technologií.

objevit vůbec, stačilo by výuka algoritmizace. V programu pro SŠ už se zcela jistě programování objevit musí, nejsme ale omezeni konkrétním programovacím jazykem, pouze by měla obsahovat průpravu jak do strukturovaného tak objektově orientovaného paradigmatu, . Pregraduální příprava z pohledu RVP by také měla obsahovat jak průpravu do paradigmatu strukturovaného programování,tak aby byl absolvent schopen zajistit výuku na střední odborné škole oboru informatika. Takováto pregraduální příprava by ale samozřejmě nedávala studentům VŠ dostatečný vhled do problematiky a nepřipravila by je na možné budoucí změny v RVP. Můžeme ale konstatovat, že podoba RVP dává jistý prostor fakultám pří přípravě programů pro budoucí učitele informatiky. Struktura těchto programů tak může reflektovat pohled fakulty na to, jak podle ní vypadá "kvalifikovaný učitel informatiky"

2.4.1 Deklarované kurikulum výuky informatiky na Slovensku

Jelikož je informatika mladý obor a změny v něm opravdu rychlé, kurikulární dokumenty nemusí obsahovat nejaktuálnější trendy v oboru. Analyzujme proto zakotvení programování ve slovenských kurikulárních dokumentech. Štátný vzdělávací program je tomu českému velmi podobný, vznikl ale o trochu později. Porovnání těchto kurikulárních dokumentů pro úroveň ISCED 1 a 2 provedl Berki (2011), nedostatečná specifikace vzdělávacího obsahu a absence práce s informatikou jako vědou a algoritmickým myšlením byly jeho hlavní zjištění. (Berki 2011, s. 36) Obsah učiva informatiky je pro školy úrovně ISCED 1-3 rozdělen do pěti okruhů:

- Informácie okolo nás
- Komunikácia prostredníctvom IKT
- Postupy, riešenie problémov, algoritmické myslenie
- Princípy fungovania IKT
- Informačná spoločnosť

Algoritmizaci se věnuje okruh Postupy, riešenie problémov, algoritmické myslenie, ve kterém jsou dále definovány obsahové a výkonové standardy[4]. Uvádím tabulku (2.2) pro úroveň ISCED3. Slovenské deklarované kurikulum má tedy pro algoritmizaci vlastní okruh, ve kterém jsou je učivo i výstupní standardy rozepsány mnohem detailněji než v kurikulu českém. U programovacího jazyka jasně uvádí, které pojmy by měl žák znát. Stejně ale jako české RVP dává prostor k případné variaci, nedefinuje paradigma programovacího jazyka ani blíže nespecifikuje programovací jazyky. Jelikož je dodržena jednotná struktura napříč všemi stupni vzdělávání, můžeme snadno identifikovat posuny v úrovni vzdělávání pro jednotlivá témata. (Berki 2016, s. 85)

Slovensko ale v inovaci pokračovalo a v roce 2015 nasadilo Inovovaný štatný vzdělávací program, který všechny oblasti ještě dále specifikuje. Uveďme nyní jako

[4]Obsahovým standardem rozumíme obsah probíraného učiva, výkonovým standardem jsou výstupní kompetence absolventa

Tabulka 2.2: Obsahový okruh *Postupy, riešenie problémov, algoritmické myslenie* z ŠVP 2011

9.citace

Výkonový štandard	Obsahový štandard
• Analyzovať problém, navrhnúť algoritmus riešenia problému, zapísať algoritmus v zrozumiteľnej formálnej podobe, overiť správnosť algoritmu. • Riešiť problémy pomocou algoritmov, vedieť ich zapísať do programovacieho jazyka, hľadať a opravovať chyby. • Rozumieť hotovým programom, určiť vlastnosti vstupov, výstupov a vzťahy medzi nimi, vedieť ich testovať a modifikovať. • Riešiť úlohy pomocou príkazov s rôznymi obmedzeniami pouţitia príkazov, premenných, typov a operácií. • Používať základné typy používaného programovacieho jazyka • Rozpoznať a odstrániť syntaktické chyby, opraviť chyby vzniknuté počas behu programu, identifikovať miesta programu, na ktorých môže dôjsť k chybám počas behu programu.	• Problém. Algoritmus. Algoritmy z beţného života. Spôsoby zápisu algoritmov. • Etapy riešenia problému – rozbor problému, algoritmus, program, ladenie. • Programovací jazyk – syntax, spustenie programu, logické chyby, chyby počas behu programu. Pojmy – príkazy (priradenie, vstup, výstup), riadiace štruktúry (podmienené príkazy, cykly), premenné, typy , množina operácií.

příklad opět tabulku s obsahovými a výkonovými standarty předmětu informatika pro úroveň ISCED3 okruhu Postupy, riešenie problémov, algoritmické myslenie pro 5.–8. ročník víceletého gymnázia[5]:**10.tabulka pro tabulku, bud využít nebo smazat či zkrátit**

[5]Na rozdíl od dokumentu z roku 2011 je obsah vzdělávání rozdělen na dva celky, 1.-4. a 5.-8. ročník

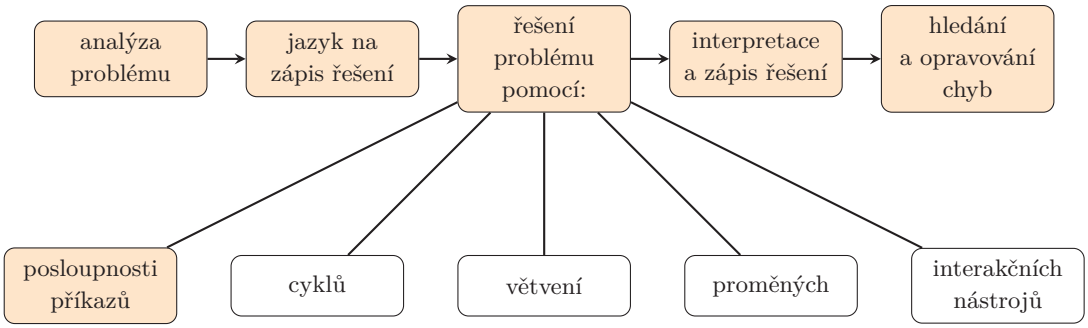
Tabulka 2.3: Obsahový okruh *programování a vývoj aplikací*

Výkonový štandard	Obsahový štandard
Analýza problému	
Žiak vie/dokáže • identifikovať vstupné informácie zo zadania úlohy, • popísať očakávané výstupy, výsledky, akcie, • identifikovať problém, ktorý sa bude riešiť algoritmicky, • formulovať a neformálne (prirodzeným jazykom) vyjadriť ideu riešenia, • uvažovať o vlastnostiach vykonávateľa (napr. korytnačka, grafické pero, robot, a pod.), • naplánovať riešenie úlohy ako postupnosť príkazov vetvenia a opakovania	Vlastnosti a vzťahy: zadaný problém – vstup – výstup Procesy: rozdelenie problému na menšie časti, syntéza riešenia z riešení menších častí, identifikovanie opakujúcich sa vzorov, identifikovanie miest pre rozhodovanie sa (vetvenie a opakovanie), identifikovanie všeobecných vzťahov medzi informáciami • význam, prvky algoritmu
Jazyk na zápis riešenia	
• používať jazyk na zápis algoritmického riešenia problému (použiť konštrukcie jazyka, aplikovať pravidlá jazyka), • rozpoznať a odstrániť chyby v zápise, • vytvárať zápisy a interpretovať zápisy podľa nových stanovených pravidiel (syntaxe) pre zápis algoritmov.	Pojem: program, programovací jazyk Vlastnosti a vzťahy: zápis algoritmu a vykonanie programu, vstup – vykonanie programu – výstup/akcia Procesy: zostavenie programu, identifikovanie, hľadanie, opravovanie chýb
Pomocou postupnosti príkazov	
• riešiť problém skladaním príkazov do postupnosti, • aplikovať pravidlá, konštrukcie jazyka pre zostavenie postupnosti príkazov.	Pojmy: príkaz, parameter príkazu, postupnosť príkazov Vlastnosti a vzťahy: ako súvisia príkazy a výsledok realizácie programu Procesy: zostavenie a úprava príkazov, vyhodnotenie postupnosti prí- kazov, úprava sekvencie príkazov (pridanie, odstránenie príkazu, zmena poradia príkazov)
Pomocou nástrojov na interakciu	
• rozpoznávať situácie, kedy treba získať vstup, • identifikovať vlastnosti vstupnej informácie (obmedzenia, rozsah, formát), • rozpoznávať situácie, kedy treba zobraziť výstup, realizovať akciu, • zapisovať algoritmus, ktorý reaguje na vstup, • vytvárať hypotézu, ako neznámy algoritmus spracováva zadaný vstup, ak sú dané páry vstup–výstup/akcia.	Vlastnosti a vzťahy: prostriedky jazyka pre získanie vstupu, spracovanie vstupu a zobrazenie výstupu Procesy: čakanie na neznámy vstup – vykonanie akcie – výstup, ná- sledný efekt

Pomocou premenných	
• identifikovať zo zadania úlohy, ktoré údaje musia byť zapamätané, resp. sa menia (a teda vyžadujú použitie premenných), • riešiť problémy, v ktorých si treba zapamätať a neskôr použiť zapamätané hodnoty vo výrazoch, • zovšeobecňovať riešenie tak, aby fungovalo nielen s konštantami	Pojmy: premenná, meno (pomenovanie) premennej, hodnota premennej, operácia (+, -, *, /) Vlastnosti a vzťahy: pravidlá jazyka pre použitie premennej, meno premennej – hodnota premennej Procesy: nastavenie hodnoty (priradenie), zistenie hodnoty (použitie premennej), zmena hodnoty premennej, vyhodnocovanie výrazu s premennými, číslami a operáciami
Pomocou cyklov	
• rozpoznávať opakujúce sa vzory, • rozpoznávať, aká časť algoritmu sa má vykonať pred, počas aj po skončení cyklu, • riešiť problémy, v ktorých treba výsledok získať akumulovaním čiastkových výsledkov v rámci cyklu • riešiť problémy, ktoré vyžadujú neznámy počet opakovaní, • riešiť problémy, v ktorých sa kombinujú cykly a vetvenia, • stanovovať hranice a podmienky vykonávania cyklov.	Pojmy: opakovanie, počet opakovaní, podmienka vykonávania cyklu, telo cyklu Vlastnosti a vzťahy: ako súvisí počet opakovaní s výsledkom, čo platí po skončení cyklu Procesy: vyhodnotenie hraníc/podmienky cyklu, vykonávanie cyklu
Pomocou vetvenia	
• rozpoznávať situácie a podmienky, kedy treba použiť vetvenie, • rozpoznávať, aká časť algoritmu sa má vykonať pred, v rámci a po skončení vetvenia • riešiť problémy, ktoré vyžadujú vetvenie so zloženými podmienkami (s logickými spojkami), • riešiť problémy, v ktorých sa kombinujú cykly a vetvenia.	Pojmy: vetvenie, podmienka Vlastnosti a vzťahy: pravda/nepravda – splnená/nesplnená podmienka Procesy: zostavovanie a upravovanie vetvenia, vytvorenie podmienky a vyhodnotenie podmienky s negáciami a logickými spojkami (a, alebo)
Interpretácia zápisu riešenia	
• krokovať riešenie, simulovať činnosť vykonávateľa s postupnos- ťou príkazov, s výrazmi a premennými, s vetvením a s cyklami, • vyjadrovať ideu daného návodu (objavovať a vlastnými slovami popísať ideu zapísaného riešenia – ako program funguje, čo zápis realizuje pre rôzne vstupy), • upraviť riešenie úlohy vzhľadom na rôzne dané obmedzenia, • dopĺňať, dokončovať, modifikovať rozpracované riešenie, • hľadať vzťah medzi vstupom, algoritmom a výsledkom, • uvažovať o rôznych riešeniach, navrhovať vylepšenie.	Vlastnosti a vzťahy: jazyk - vykonanie programu Procesy: krokovanie, čo sa deje v počítači v prípade chyby v programe

Hľadanie a opravovanie chýb	
• rozpoznávať, že program pracuje nesprávne, • hľadať chybu vo vlastnom, nesprávne pracujúcom programe a opraviť ju, • zisťovať, pre aké vstupy, v ktorých prípadoch, situáciách program zle pracuje, • uvádzať kontra príklad, kedy niečo neplatí, nefunguje, • posúdiť a overiť správnosť riešenia (svojho aj cudzieho), • rozlišovať chybu pri realizácii od chyby v zápise.	Vlastnosti a vzťahy: chyba v postupnosti príkazov (zlý príkaz, chýbajú- ci príkaz, vymenený príkaz alebo príkaz navyše), chyba vo výrazoch s premennými, chyba v algoritmoch s cyklami a s vetvením, chyba pri realizácii (logická chyba), chyba v zápise (syntaktická chyba) Procesy: rozpoznanie chyby, hľadanie chyby

Výhodou je, že toto dělení je zachováno pro všechny stupně vzdělávání, snadno se identifikuje posun žáků v jejich znalostech a dovednostech jak píše Berki (2016, s. 85), který uvádí i tabulku posunu pro jednotlivé celky napříč celky. Pro náš celek Algoritmické řešení problému je zajímavé také to, že úzce kopíruje samotný algoritmus vývoje softwaru, můžeme ho tedy zobrazit v přehledném diagramu (obrázek2.3), barevně jsou označené bloky společné pro všechny úrovně vzdělávání:



Obrázek 2.3: Obsahový okruh *programování a vývoj aplikací*

Slovenské deklarované kurikulum podobně jako to české nespecifikuje konkrétní programovou výbavu nebo programovací paradigma. Na rozdíl od něj ale specifikuje a rozděluje obsah vzdělávání do přehledných a smysluplných kategorií, úroveň dosaženého vzdělání je tím lépe ověřitelná.

Pokud budeme předpokládat, že jedním z determinantů pregraduální přípravy jsou i deklarované kurikulum na národní úrovni, slovenská verze kurikula má se svoji specifikovanější podobou výhodu. Tím, že slovenské kurikulum vymezilo pro algoritmizaci a programování vlastní okruh, dává tomuto tématu vysokou důležitost a bere ho jeho nedílnou součást výuky informatiky na základních a středních školách.

11.zkusit zahrnout i Brity

2.5 Proč učit programování?

12.Chybí citace, je potřeba trošku uhladit Programování je běžně součástí pregraduální přípravy učitelů (Berki 2013), toto téma ale nemá u studentů velkou oblibou (Fojtík 2015). Studenti mají obtíže předměty zaměřené na programování absolvovat,"a proto by se raději ve své budoucí pedagogické praxi výuce programování vyhnuli." (Fojtík 2015, s. 54) Studenti se staví spíše proti výuce programování na základních školách, má být podle nich součásti jen výběrových předmětů na gymnáziích. Uvádí důvody jako:

- Výuka programování je příliš složitá pro žáky základních i většiny středních škol,
- programovací jazyky jsou nesrozumitelné a žáci by je nezvládli,
- vývojové nástroje jsou komplikované a nepřehledné,
- výuka programování je časově náročná a nezůstal by čas na důležitější témata,
- studenti mají negativní zkušenost s výukou programování na střední nebo základní škole,
- nepotřebujeme tolik programátorů v praxi.

Takovýmto studentům je potřeba objasnit několik důvodů, proč by mohlo a mělo být programování a algoritmizace součástí kurikula jak na vysoké škole tak na nižších stupních. (Fojtík 2015, s. 54) Výuka programování a algoritmizace není příprava na povolaní programátora, ale „Cílem programování ve škole je rozvoj tvořivosti a myšlení . Samotné programování je (skvělým) nástrojem k dosahování těchto cílů." (Lessner 2015) Koncepty které se studenti naučí během odborného výkladu mohou na základních a středních školách předávat způsobem, kterému rozumí děti, např. Pomocí dětských programovacích jazyků jako Scratch nebo Logo, případně programovat jednoduché roboty.

Schubert a Schwill (Schubert & Schwill 2011) označili jako základní myšlenky informatiky pojmy algoritmus, jazyk a strukturální rozklad. Programování s těmito pojmy úzce souvisí, dá se tak označit za jedno z důležitých témat informatiky a mělo by tak být součástí oboru, který připravuje budoucí učitele informatiky

Ačkoliv v našem RVP má algoritmizace a programování jen malé zastoupení oproti informačním technologiím zaměřeným uživatelským způsobem, v zahraničí se situace v poslední dobou mění. „V současné době probíhají v řadě zemí kurikulární reformy, v nichž se vymezuje a mění postavení informatiky. Kurikulární reforma, v níž má významné postavení informatická složka, probíhá například v Polsku, v Austrálii nebo v Rusku." (Černochová & Vaníček 2015). Na Slovensku se setkávají žáci s informatickými tématy jako algoritmické myšlení procedury a princip fungování digitálních technologii už od 3. třídy. V Anglii výuka informatiky probíhá ve specializovaném předmět Computing,(cas, didaktika inf. Na startu) jehož cílem je rozumět a aplikovat základní principy informatiky, kdy se už žáci na prvním stupni základních škol učí psát jednoduché programy. (Department for Education 2013) Je možné, že se dočkáme v ČR změny kurikulárních dokumentů tímto směrem, učitelé by na to měli být připraveni.

Velkým příznivcem programování ve vzdělávání byl Seymour Papert, tvůrce

dětského programovacího jazyku LOGO. Ve své knize Mindstorms (Papert 1993)popisuje mnoho výhod, které mají počítače a programování na vývoj a vzdělávání dětí. Ačkoliv tato kniha vyšla už v roce 1980, můžeme některé myšlenky označit jako nadčasové[6]. Podle Paperta někteří žáci mají model učení postavené na schématu, ve kterém je výsledek špatně nebo správně, neexistuje jiná možnost. Ale v programování většinou prvotní verze programu není správně, je potřeba najít a opravit chyby. Při tomto schématu pak není hlavní otázkou zda je program správně nebo špatně, ale zda je opravitelný. Další výhodou je, že při programování žáci využívají a učí se matematickému jazyku a matematice, ke kterému mají lepší vztah, protože je pro ně matematika využitelný nástroj, ne cíl výuky.

2.6 Jak učit programování?

Zaměřme se nyní, na to jakým způsobem je možné programování vyučovat. "Výuka algoritmizace a programování prochází v současné době velkými změnami, které se snaží reagovat na dynamický rozvoj softwarového průmyslu. Dříve využívané metodické postupy, modely vývoje či programovací jazyky nedostačují aktuálním potřebám. (FOJTIK 2013)

Většinou probíhá výuka nejprve pomocí procedurálního paradigmatu, během kterého jsou studenti obeznámeni s datovými typy, vytvářením proměnných, operátorech, cyklech, podmínkách apod. Výuka je úzce spojena se syntaxí jazyka, bez které by se student neobešel. Až následně se přechází k výuce objektového programování a pojmům jako třída objekt dědičnost. Jednou z nevýhod takového přístupu je, že „žáci z tohoto postupu získají pocit, že objektové programování je jen určitá nadstavba jazyka". (FOJTIK 2013) Výuka může probíhat i pomocí jiných metodik, představme si krátce některé z nich:

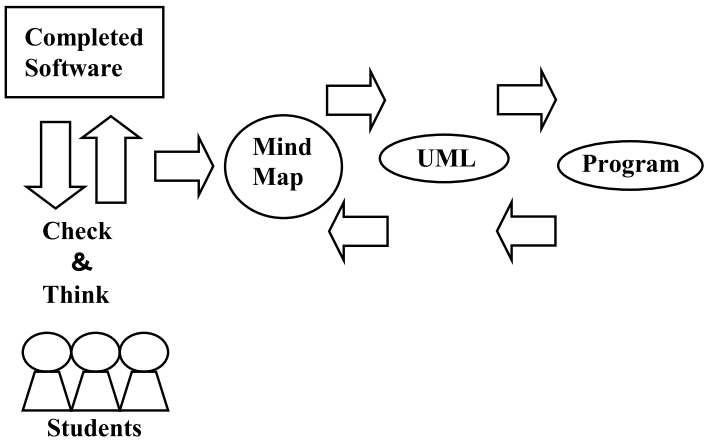
Object first Filozofie této metodiky se snaž studentovi v první řadě představit důležité koncepty objektově orientovaného programování **13.citace - stránky blueJ**, aniž by se musel zabývat syntaxí programovacího jazyka. Autoří této metodiky vytvořili vývojové prostředí BlueJ pro objektově orientovaný programovací jazyk Java. Toto prostředí je jednoduché na ovládání a dokáže vizualizovat třídy programu do diagramu.

Architecture first Tato metodika se snaží rozšiřovat pojetí Object first, ke které přidává ještě větší důslednost na znalost architektury softwaru, než se přejde k samotnému kódování programu.

Algorithm first metodika se snaží zaměřit více pozornosti na vytváření algoritmů, než na jejich následném kódování. Studenti tráví více času navrhováním algoritmu a jejich vizualizaci pomocí vývojových diagramů. Díky tomu by měli být schopni převést problémy reálného světa do podoby algoritmu.

[6]Pepert úzce spolupracoval i s Jeanem Piagetem, tvůrcem známe teorie kognitivního vývoje, která se na pedagogických fakultách učí dodnes

Agilní metodiky Tento pojem zastřešuje přístup k vývoji softwaru pomocí několika principů např. ”Nejúčinnějším a nejefektnějším způsobem sdělování informací vývojovému týmu z vnějšku i uvnitř něj je osobní konverzace.” nebo ”Hlavním měřítkem pokroku je fungující software”. Studenti v tomto přístupu zastávaji roli programátorů, vzájemně diskutují a pracují na návrhu softwaru, ke kterému požívají myšlenkové mapy a následně UML diagramy. Poté se přistupuje k samotnému programování, při kterém se snaží napsat alespoň malou ale hlavně funkční část kódu, což by je mělo motivovat do další práce. Případné problémy mohou vyřešit v kooperaci s ostatními.



Obrázek 2.4: Proces provedení algoritmu počítačem (Harel & Feldman 2004, s. 56)

Game first Tato metodika si klade jako prioritu zaujmout studenty. Výuka probíha formou vývoje počítačové hry, při kterém by se měli studenti seznámit i z odbornými znalostmi programování.

Důležitým aspektem ve výuce programováním je nejen volba paradigmatu a metodiky, ale i vhodného programovacího jazyka. Milebrandt definoval několik vlastností, které by měl mít programovací jazyk vhodný pro vzdělávání jako například:

- jednoduchost použití
- jednoduchá syntaxe
- dobré testovací nástroje
- smysluplné názvy klíčových slov

Některé další aspekty definovala Manilla, např.

- je zdarma
- má širokou podporující komunitu
- jeho součástí je dobrý výukový materiál
- je rozšiřitelný …

Uveďme si nyní několik programovacích jazyků, které jsou v současnosti relevantní ve výuce programování. Jedná se o jazyky, které jsou v českém prostředí

známé (články o nich se objevují ve sbornících odborných konferencí) a jsou vhodné pro výuku programování jak uvádí ve svém výzkumu Manilla (2006).

Python Jazyk Python byl poprvé vydán v roce 1991, jeho autorem je Guido Van Rossum. Tento jazyk je interpretovaný a multiparadigmatický, obsahuje konstrukce objektově orientovaného, funkcionálního i imperativního paradigmatu. Jeho výhodou je přehlednost, byl vytvořen s důrazem na výuku programování. Využívá se k výuce na předních amerických univerzitách(2014). , může být použit i k výuce na základních školách. "Jazyk sa rýchlo a dobre učí. Programovanie je pragmatické a rýchlo vedie k stručnému a efektívnemu programovému kódu." (Hájek & Peter 2015)

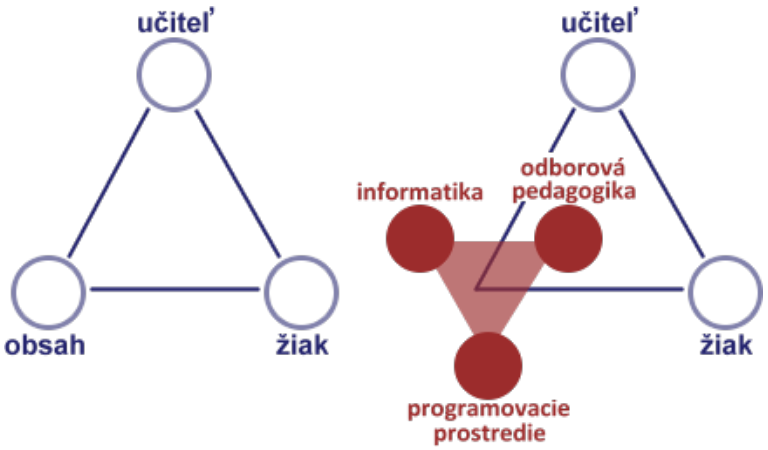
Java Java Je objektově orientovaný programovací jazyk, vyvinutý v roce 1990 Jamesem Gasolinem ve firmě Sun Microsystem. Jeho syntaxe vychází z populárních jazyků C a C++ (Gosling et al. 2014). Hojně se využívá i k výuce programování na vysokých školách. Její výhodou je značná nezávislost na zařízení, je tak využívána např. na mobilních zařízeních jako mobilní telefony nebo tablety. Pro Javu vzniklo několik vývojových prostředí vhodných pro výuku programování, které umožňuj vizualizovat jednotlivé třídy a objekty programu jako např BlueJ (Pecinovský 2004) nebo Greenfoot (Greenfoot n.d.).

Scratch Programovací jazyk Scratch je speciálně vytvořen pro výuku dětí. Patří mezi vizuální programovací jazyky, programování tak probíhá manipulací s grafickými prvky. I přes svoji jednoduchost dovoluje vytvořit i složitější programátorské konstrukce. Další jeho výhodou je možnost jednoduše sdílet vytvořené projekty v komunitě. Ačkoliv se nejedná o klasický, komerčně používaný jazyk, je vhodný k představení základů a vizualizaci konceptů programování.(MUSILEK 2013)

Scheme Jazyk Scheme vznikl v roce 1975, jeho autoři jsou Guy Lewis Steele a Gerald Jay Sussman. Vychází z programovacího jazyka Lisp, je ale značně zjednodušen, "je možné syntaxi jazyka Scheme úplně vyložit během jedné přednášky" (Skoupil 2007, s. 11)". Jedná se o multiparadigmatický jazyk, "Jeho podstatou je programovací paradigma funkcionální, je v něm ale možné pracovat imperativním nebo objektově orientovaným stylem. Scheme je proto často označován jako „algoritmický" jazyk, tedy jazyk, ve kterém je možno snadno formulovat algoritmy. (Skoupil 2007, s. 10)"

V současnosti může výuka programování probíhat za pomoci mnoha různých programovacích jazyků ,vývojových prostředí a dalších nástrojů např. pro sdílení hotových programů. Každý jazyk, prostředí i nástroj však může mít jiný dopad na rozvoj gramotnosti člověka, jak zmiňují Proctor a Blikstein (Proctor & Blikstein 2016). Ti vytvořili tři kategorie, do kterých je jednotlivé jazyky a nástroje možno rozdělit (zestručněno):

- materiální – nástroje, které se soustřeďují hlavně na výuku práce s kódem
- kognitivní – nástroje, které napomáhají rozvoji rekurzivního a informatického myšlení a abstrakce



Obrázek 2.5: Proces provedení algoritmu počítačem (Harel & Feldman 2004, s. 56)

- sociální – nástroje, které dovolují využívat kód vytvořený někým dalším nebo se zapojovat do různých komunit

Je důležité, aby student učitelství informatiky uvědomoval široké možnosti, jak na své žáky pomocí programování působit a pokud možno udržoval jednotlivé aspekty v rovnováze, což zmiňuje ve své práci i Kalaš (Kalaš 2016), který vytvořil rozšířenou verzí didaktického trojúhelníka, ve které rozšiřuje vrchol obsahu na tři podvrcholy – informatika, oborová didaktika a programovací prostředí. Tyto tři složky by měly být ve výuce stejně zastoupeny, jinak by mohlo podle autora dojít k deformacím. Pokud se obsah zaměřuje hlavně na programovací prostředí, může to vést k "technocentrizmu", pokud převládá informatická složka, Kalaš ji doslova označuje jako "Computer Sience centrizmus", což popisuje jako přehnanou snahu zpopularizovat obor informatika na úkor zbylých dvou složek. Nakonec preference didaktického obsahu může vést k "plytkému nebo žádnému porozumění programovacích konstruktů".**14.Dopsat že kvůli tomu by určitě měla být součástí didaktika informatiky**

Pokud bereme v potaz vše výše uvedené, je zřejmé, že teorie výuky programování dosahuje značné šíře. Aby si byl budoucí učitel informatiky mohl být vědom možností a potencionálních problémů, které výuka programování může představovat, považuji za nutné, aby studenti byly obeznámeni s didaktikou programování a to nejlépe v samostatném předmětu. Jen tak budou moci studenti chápat i další souvislosti výuky programování, protože například jak píše Lessner (LESSNER 2013, s. 13) "Programování jako takové (vývoj softwaru, popř. zápis algoritmů) ale na gymnáziu těžko obstojí v roli vzdělávacího cíle. Tímto cílem může být kultivace myšlení, rozvoj schopnosti systematicky analyzovat a řešit problémy, nikoliv znalost konkrétního programovacího jazyka. Proto by i výuka programování by měla brát v první řadě důraz na pochopení algoritmů před důkladnou znalostí syntaxe jazyka. **15.mozno se ještě dale oprit o clanek Kalase o didkatice programování z Didinfa 2009**

3 Analýza deklarovaného kurikula pregraduální přípravy

Příprava budoucích učitelů informatiky probíhá v ČR na přírodovědných a a pedagogických fakultách v rámci strukturovaných dvoustupňových programů. V rámci praktického výzkumu byla nejdříve provedena rešerše zdrojů týkajících se této problematiky, následně byla proveden vlastní výzkum skládající se z vyhledání všech dostupných programů pro pregraduální přípravu učitelů informatiky, vyhledání a kategorizace předmětů zaměřených na programování a algoritmizaci a jejich následná textová analýza. V každé části výzkumu je popsána příslušná metodika.

3.1 Stav problematiky

V roce 2013 analyzoval programy pro pregraduální přípravu učitelů informatiky Berki. Byla provedeno porovnání akreditovaných programů hlavně z hlediska poměru počtu kreditů pro předem definované kategorie (matematika, algoritmy, databázové a operační systémy, publikační systémy, technologie počítačů, didaktika). Podle této studie bakalářské programy zaměřují spíše na odbornou informatiku, zatímco navazující magisterské programy se zaměřují spíše na didaktiku informatiky.

3.2 Pregradualní příprava učitelů informatiky – popis zkoumaného vzorku

Předmětem zkoumání byly studijní programy zaměřené na pregraduální přípravu učitelů informatiky na základních a středních školách. Studie se zaměřuje čistě na prezenční studium, nejsou jde zahrnutý kombinované ani dálkové programy. Pro relevantnost studie byly do výzkumu zahrnuty pouze programy, do kterých jsou přijímáni studenti pro rok 2017/2018. Tímto opatřením bylo tak zajištěno, že budou zahrnuty pouze programy s platnou akreditací. V ČR najdeme i některé programy, které spojují informatiku a technickou výchovu, které jsou zaměřeny hlavně pro budoucí učitele odborných středních škol. Tyto programy se značně svým obsahem liší, proto nejsou do studie zahrnuty. Jelikož mohou vysoké školy akreditovat programy vzdělávající budoucí učitele informatiky akreditovat pod různými jmény, byla provedena rešerše otevíraných oborů na všech fakultách

vzdělávajících učitele v ČR. Z této rešerše vznikl vzorek 25 studijních programů ze 14 fakult 10 univerzit:**16.dát oficální názvy nebo nechat názvy s městy?????**

- Jihočeská univerzita v Českých Budějovicích (JU) – Pedagogická fakulta (PedF), Přírodovědecká fakulta (PřF)
- Masarykova univerzita v Brně (MU) – Fakulta informatiky (FI)
- Ostravská univerzita – Pedagogická fakulta (PedF), Přírodovědecká fakulta (PřF)
- Technická univerzita v Liberci (TUL) – Fakulta přírodovědně-humanitní a pedagogická (FPHP)
- Univerzita Hradec Králové (UHK) – Přírodovědecká fakulta (PřF), Pedagogická fakulta (PedF)
- Univerzita Jana Evangelisty Purkyně v Ústí nad Labem (UJEP) – Přírodovědecká fakulta (PřF)
- Univerzita Karlova v Praze (UK) – Matematicko-fyzikální fakulta (MFF), Pedagogická fakulta (PedF)
- Univerzita Palackého v Olomouci (UP) – Přírodovědecká fakulta (PřF)
- Univerzita Tomáše Bati ve Zlíně (UTB) – Fakulta aplikované informatiky (FAI)
- Západočeská univerzita v Plzni (ZČU) – Fakulta pedagogická (PedF)

Pro potřeby studie bylo potřeba vybrat předměty, které souvisí s výukou programování. Na základě zkoumání složení předmětů v jednotlivých programech bylo vytvořeno 5 skupin předmětů, jenž mají souvislost s programováním a algoritmizací. Jelikož ze sylabů předmětů se nedá zjistit časová dotace, která je jednotlivým tématům věnována, do jednotlivých kategorii byl předmět zařazen v případě, pokud se v něm vyskytuje níže popsané téma a to v jakékoliv míře. Pokud sylabus předmětu obsahuje témata z více kategorií, je zařazen do té kategorie, ze které jeho sylabus obsahuje více témat.

- **Výuka programování** – do této kategorie byly zařazeny všechny předměty, jejímž obsahem je výuka programování či teorie přímo související s programování jako je teorie paradigmat programování.
- **Teoretická informatika** – do této kategorie je zařazen předmět, pokud alespoň částí jeho obsahu je teorie informatiky, která s programováním souvisí. Jedná se o teorii automatů a formálních jazyků, gramatiky, složitosti a vyčíslitelnosti.
- **Teorie algoritmů** – Do této kategorie je zařazen předmět, pokud je alespoň částí jeho obsahu teorie algoritmů.
- **Programování v prostředí WWW** – do této kategorie jsou zařazeny všechny předměty, které jsou primárně zaměřeny na tvorbu webových stránek a jejichž součástí je programování v jazyku JavaScript nebo PHP, případně jiného .

- **Didaktické předměty** – do této kategorie je zařazen předmět, pokud alespoň částí jeho obsahu je didaktika programování nebo didaktika algoritmizace.

Co je možné porovnat:

- Porovnání programů hlavně z hlediska programování tedy:
  - porovnání množství předmětů zaměřených na programování
  - porovnání používaných programovacích jazyků
  - porovnání umístění předmětu v rámci studia
- Porovnání didaktiky programování
  - Její výskyt
  - její umístění v rámci studia
  - Její umístění vůči předmětům zaměřených na programování

uni.	fakulta	typ	program
JU	Pedagogická fakulta	Bc.	Informační technologie se zaměřením na vzdělávání
		NMgr.	Učitelství informatiky pro 2. stupeň základních škol
	Přírodovědecká fakulta	Bc.	Informatika pro vzdělávání
		NMgr.	Učitelství informatiky pro střední školy
MU	Fakulta informatiky	Bc.	Informatika a druhý obor
		NMgr.	Učitelství informatiky pro střední školy
OU	Přírodovědecká fakulta	Bc.	Informatika (dvouoborové)
		NMgr.	Učitelství informatiky pro 2. stupeň základních škol a střední školy
	Pedagogická fakulta	Bc.	Informační a komunikační technologie se zaměřením na vzdělávání
TUL	Přírodovědně-humanitní a pedagogická	Bc.	Informatika se zaměřením na vzdělávání
		NMgr.	Učitelství informatiky pro střední školy
			Učitelství informatiky pro 2. stupeň základní školy
UHK	Přírodovědecká fakulta	Bc.	Informatika se zaměřením na vzdělávání
	Pedagogická fakulta	NMgr.	Učitelství pro 2. stupeň ZŠ - informatika
			(Učitelství pro střední školy - informatika)
UJEP	Přírodovědecká fakulta	Bc.	Informatika
UK	Matematicko-fyzikální fakulta	Bc.	Informatika se zaměřením na vzdělávání
		NMgr.	Učitelství informatiky
	Pedagogická fakulta	Bc.	Informační technologie
		NMgr.	Informační a komunikační technologie
UP	Přírodovědecká fakulta	Bc.	Informatika pro vzdělávání
		NMgr.	Učitelství informatiky pro střední školy
UTB	Fakulta aplikované informatiky	NMgr.	Učitelství informatiky pro střední školy
ZČU	Fakulta pedagogická	Bc.	Informatika se zaměřením na vzdělávání
		NMgr.	Učitelství informatiky pro ZŠ

Tabulka 3.1: Počet hodin programování v bakalářských programech

PedF JU	PřF JU	FI MU	PřF OU	PedF OU	FP TUL	PřF UHK	PřF UJEP	MFF UK	PedF UK	PřF UP	PedF ZČU
1	2	3	1	2	3	3	2	4	2	4	2

* Blahblah

Tabulka 3.2: Počet hodin programování v NMgr. programech

PedF JU	PřF JU	FI MU	PřF OU	FP TUL ZŠ	FP TUL SŠ	PedF UHK ZŠ	PedF UHK SŠ	MFF UK	PedF UK	PřF UP	FAI UTB	PedF ZČU
0	0	0	0	2	2	0	0	0	0	0	0	0

Tabulka 3.3: Počet didaktických předmětůí v bakalářských programech

PedF JU	PřF JU	FI MU	PřF OU	PedF OU	FP TUL	PřF UHK	PřF UJEP	MFF UK	PedF UK	PřF UP	PedF ZČU
2	0	0	0	0	0	0	0	0	1	0	0

Tabulka 3.4: Počet didaktických předmětů v NMgr. programech

PedF JU	PřF JU	FI MU	PřF OU	FP TUL ZŠ	FP TUL SŠ	PedF UHK ZŠ	PedF UHK SŠ	MFF UK	PedF UK	PřF UP	FAI UTB	PedF ZČU
2	3	2	1	1	2	0	0	2	1	2	3	2

4 Textová analýza

4.1 Popis vzorku

4.2 Popis způsobu analýzy

4.3 Výsledky

17.Srovnat s kapitolou 3.1 18.Ještě tři podkapitoly:1Jaká je dotace předmětů Jaká je jejich struktura (jak na sebe navazují, jak jsou oddělené, jestli jsou algoritmy zvlašt nebo dohromady s programováním) Obsah předmětů

5 Návrh konceptu pregraduální přípravy

5.1 ZŠ

5.2 SŠ

6 Závěr

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut

metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Literatura

Berki, J. (2011), Ict in the czech and slovak national curriculum, in 'Proceedings of 5th Internaitonal Conference ISSEP', Association of the Infovek Project and Faculty of Mathematics, Physics and Informatics, Comenius University, Bratislava.

Berki, J. (2013), Study programs of training teachers of informatics in the czech republic, in 'Journal of Technology and Information Education', Vol. 5.
URL: *http://www.jtie.upol.cz/clanky_1_2013/JTIE-1-2013.pdf*

Berki, J. (2016), Projektované, realizované a dosažené ICT kurikulum na základních školách, Disertační práce, Jihočeská univerzita v Českých Budějovicích, Pedagogická fakulta, České Budějovice.
URL: *http://theses.cz/id/y5olfe/*

Bolshakova, E. (2005), 'Programming paradigms in computer science educaion', International journal information theories .
URL: *http://www.foibg.com/ijita/vol12/ijita12-3-p13.pdf*

Cormen, T. H. (2013), Algorithms Unlocked, MIT University Press Group Ltd.

Cormen, T. H., Leiserson, C. E., Rivest, R. L. & Stein, C. (2009), Introduction to Algorithms, The MIT Press.

Department for Education (2013), The National Curriculum in England: Key Stages 1 and 2 framework document. Available at: https://www.gov.uk/government/publications/national-curriculum-in-england-primary-curriculum (Accessed: 6 December 2016).

Černochová, M. & Vaníček, J. (2015), Didaktika informatiky na startu, in 'Oborové didaktiky : vývoj – stav – perspektivy', Masarykova univerzita, Brno.

FOJTIK, R. (2013), 'Moderní přístupy k výuce programování', JTIE **5**(1), 58–62.
URL: *http://jtie.upol.cz/cz/artkey/jti-201301-0008.php*

Fojtík, R. (2015), Vzdělávání budoucích učitelů informatiky, in 'Didinfo 2015'.

Gosling, J., Joy, B., Steele, G. L., Bracha, G. & Buckley, A. (2014), The Java Language Specification, Java SE 8 Edition, 1st edn, Addison-Wesley Professional.

Greenfoot (n.d.), 'About greenfoot'.
URL: *https://www.greenfoot.org/overview*

Guo, P. (2014), 'Python is now the most popular introductory teaching language at top u.s. universities'.
URL: *http://cacm.acm.org/blogs/blog-cacm/176450-python-is-now-the-most-popular-introductory-teaching-language-at-top-u-s-universities*

Harel, D. & Feldman, Y. (2004), Algorithmics: The Spirit of Computing (3rd Edition), Addison-Wesley.

Hájek, Z. & Peter, V. (2015), Skúsenosti s programovaním na základnej škole, in 'Didinfo 2015'.

Kalaš, I. (2016), Scratchmaths: vzdelávací obsah a princípy tvorby, in 'Spojená medzinárodná konferencia DidInfo&DidactIG 2017', UMB Banská Bystrica, pp. 16–24.

Škavrda, L. (2006), 'Co je to funkcionální programování'. http://programujte.com/clanek/2006032503-co-je-to-funkcionalni-programovani/.

Knuth, D. E. (2008), Umění programování. 1. díl: Základní algoritmy, Computer Press.

LESSNER, D. (2013), Vyuka efektivity algoritmu na gymnaziich, Vol. 5, pp. 12–20.
URL: *http://jtie.upol.cz/cz/artkey/jti-201301-0002.php*

Lessner, D. (2015), 'Povinné programování od první třídy — zamkněte děti doma!'.
URL: *http://ucime-informatiku.blogspot.co.id/2015/10/povinne-programovani-od-prvni-tridy.html*

Mannila, L. & de Raadt, M. (2006), An objective comparison of languages for teaching introductory programming, in 'Proceedings of the 6th Baltic Sea Conference on Computing Education Research: Koli Calling 2006', Baltic Sea '06, ACM, New York, NY, USA, pp. 32–37.
URL: *http://doi.acm.org/10.1145/1315803.1315811*

MŠMT (2016), 'Rámcový vzdělávací program pro základní vzdělávání'.
URL: *http://www.nuv.cz/uploads/RVP_ZV_2016.pdf*

Motyčka, A. (1999), Algoritmizace. Brno: Konvoj.

MUSILEK, M. (2013), 'Project scratch', JTIE **5**(1), 102–106.
URL: *http://jtie.upol.cz/artkey/jti-201301-0015.php*

Papert, S. A. (1993), Mindstorms: Children, Computers, And Powerful Ideas, Imprint unknown.

Pecinovský, R. (2004), 'Bluej – vývojové prostředí pro výuku jazyka java'.

Proctor, C. & Blikstein, P. (2016), Grounding how we teach programming in why we teach programming, in 'Constructionism in Action 2016'.

Schubert, S. & Schwill, A. (2011), Didaktik der Informatik, Spektrum-Akademischer Vlg.

Skiena, S. (2008), The Algorithm Design Manual, Springer London Ltd.

Skoupil, D. (2007), PROGRAMY A PROJEKTY V JAZYKU SCHEME I.

Seznam tabulek

2.1 Obsahový okruh *programování a vývoj aplikací* 16
2.2 Obsahový okruh *Postupy, riešenie problémov, algoritmické myslenie* z ŠVP 2011 18
2.3 Obsahový okruh *programování a vývoj aplikací* 19

3.1 Počet hodin programování v bakalářských programech 31
3.2 Počet hodin programování v NMgr. programech 31
3.3 Počet didaktických předmětůí v bakalářských programech 31
3.4 Počet didaktických předmětů v NMgr. programech 31