



TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta přírodovědně-humanitní
a pedagogická



Programování v pregraduální přípravě učitelů informatiky[verze 14. 5.]

Diplomová práce

Studijní program: N1101 – Matematika
Studijní obory: 7504T077 – Učitelství informatiky pro střední školy
7504T089 – Učitelství matematiky pro střední školy
Autor práce: **Bc. Ondřej Vraštil**
Vedoucí práce: Mgr. Jan Berki, Ph.D.





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.



Tento list nahradte
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 programování – odborné definice a vztahy	10
2.2	Vztah algoritmizace a programování	11
2.3	Programovací paradigmatata	12
2.4	Programování z pohledu deklarovaného kurikula ZŠ a SŠ	14
2.4.1	Deklarované kurikulum výuky informatiky na Slovensku . . .	16
2.5	Proč učit programování?	21
2.6	Jak učit programování?	22
3	Analýza deklarovaného kurikula pregraduální přípravy	25
3.1	Stav problematiky	25
3.2	pregradualní příprava učitelů informatiky-popis zkoumaného vzorku .	25
4	Textová analýza	28
4.1	Popis vzorku	28
4.2	Popis způsobu analýzy	28
4.3	Výsledky	28
5	Návrh konceptu pregraduální přípravy	29
5.1	ZŠ	29
5.2	SŠ	29
6	Závěr	30

1 Úvod

Nedílnou součástí pregraduální přípravy budoucích učitelů informatiky základních i středních škol je výuka programování 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 doporučeném kurikulu 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.

Nutnou podmínkou správně provedené didaktické transformace látky směrem k žákovi je nadhled a vzdělání učitele, jež by měl nabýt během svého vysokoškolského studia. 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 aktualita, které mají pro informatiku, jakožto mladý a dynamicky se rozvíjející obor velký význam. Fakulty připravující učitele informatiky by měli pružně reagovat na moderní trendy ve výuce a uspokojit poptávku po kvalifikovaných učitelích. 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. 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 programování 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 pomocí 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. ???tahle kapitola by potřebovala trochu vyladit a zpřehlednit, algoritmizaci v RVP popsala už ve své diplomové práci Bromová, lepší se odkazovat nebo přepracovat?	14
2. ???Je to kategorie, nebo se to nazývá jinak?	15
3. lépe	15
4. citace	15
5. citace	17
6. Chybí citace, je potřeba trošku uhladit	21
7. citace - stránky blueJ	22
8. Jak jsem tyto jazyky volil	23
9. http://cacm.acm.org/blogs/blog-cacm/176450-python-is-now-the-most-popular-introductory-teaching-language-at-top-u-s-universities/fulltext . .	24
10. Skúsenosti s programovaním na základnej škole,didinfo2015	24
11. Java Programming Language ,Kamran Sartipi,August, 1996	24
12. http://jtie.upol.cz/pdfs/jti/2013/01/15.pdf	24
13. http://bit.ly/2lKDFKC	24


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? 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 programování – odborné definice a vztahy

„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 (?). V literatuře najdeme několik jeho definic.  hubertová a Schwill popisují algoritmus jako formálními prostředky popsateľný, mechanicky provediteľný postup k řešení třídy problému (? , s. 4). Cormen popisuje algoritmus jako sadu kroků ke splnění úlohy, počítačový algoritmus jako sadu kroků ke splnění úlohy tak přesně popsanych, aby je dokázal vykonat počítač.(?, s. 1). Pro Skienu je algoritmus procedura ke splnění konkrétní úlohy a myšlenka, která stojí za počítačovým programem (? , s. 3). Cormen, Leiserson, Rivest¹ a Stein 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 (? , s. 5). Znamý informatik Donald Knuth, tvůrce typografického systému Tex² definuje algoritmus jako konečnou množinou pravidel, která popisují posloupnost operací pro řešení jistého typu problémů (? , s. 5). Další definici najdeme u Harela a Feldmana: Algoritmus je abstraktní návod předepisující proces, který by mohl být proveden člověkem, počítačem nebo jinými prostředky (? , s. XII). Podle Knutha (? , s. 4-6) musí algoritmus splňovat několik základní vlastností:

¹písmeno „R“ v RSA

²kterým 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 poposány 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.

Často najdeme přirovnání algoritmu ke kuchařskému receptu jako například u Knutha (?, s. 6) nebo u Harela a Feldmana (?, 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**. (?, s. 67) Motýčka 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* (?, s. 5). 

2.2 Vztah algoritmizace a programování



Pokud jsme si definovali algoritmus obecně, musíme definovat i propojení tohoto pojmu s počítači – stroji. Počítače v současné době k řešení mnoha problémů okolo nás, např. automatickému pilotování letadla, nebo složitým simulacím chemických procesů. (?, 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 (?, s. 6). V současné době existuje mnoho jazyků lišících se syntaxí a použitím. Jazyky se liší mírou abstrakce – vyšší jazyky se podobají lidské řeči, jsou více abstraktní, naopak pomocí nižších jazyků je možné lépe ovládat strojové operace, mají nižší abstrakci. Schéma (2.4) 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.



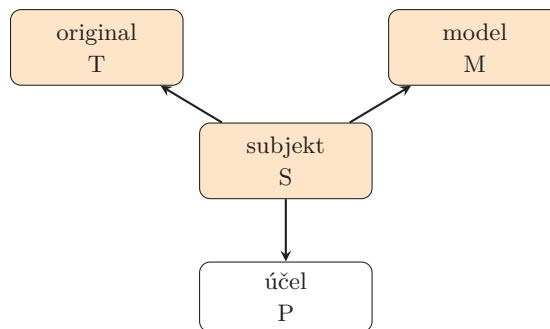
Obrázek 2.1: Proces provedení algoritmu počítačem (?, s. 56)

Alternativa ke kompilaci je převod pomocí interpretru. Na rozdíl od kompilace dokáže interpret provést program aniž by byl převeden do strojového kódu (?, s. 54-57)

2.3 Programovací paradigmatata

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) (?, 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 představují specifický přístup k programu a jeho provedení. Takovéto styly nazýváme programovací paradigmatata (?, s. 286).

Imperativní paradigma se vyvinulo z nižších jazyků – strojového a jazyku



Obrázek 2.2: Schéma vytváření modelů (?, s. 135)

symbolických adres (?, s. 286). Z pohledu toho paradigmatu je počítač soubor paměťových buněk organizovaných do různých typů datových struktur, jako pole nebo seznam. Program je 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 **Funkčním 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. není zde princip modifikovatelné paměti jako v imperativním paradigmatu (?).

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 **oběktově orientovaného paradigmatu** se zakládá na představě, že realný svět je množství objektů, které spolu navzájem komunikují. Definujeme zde skupiny prvků s podobnými vlastnostmi nazývané třídy. Na rozdíl od imperativního paradigmatu, ve kterém jsou funkce považovány za aktivního činitele modifikujícího pasivně uložená data, oběktově 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 (?, 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 theoremý	Logické dokazování theoremu	Úspěch nebo neúspěch dokazování
Objektově orientované	Objekt	Soubor tříd a objektů	Výměna zpráv mezi objekty	Finální stav objektových stavů

Programovací jazyk nemusí podporovat pouze jedno paradigma, takovým jazykům pak říkáme **multiparadigmatické**. 


2.4 Programování z pohledu deklarovaného kurikula ZŠ a SŠ



1.???tahle kapitola by potřebovala trochu vyladit a zpřehlednit, algoritmizaci v RVP popsala už ve své diplomové práci Bromová, lepší se odkazovat nebo 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. V jaké míře je zde programování nebo algoritmizace obsažena?

V RVP pro základní vzdělávání (?) 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 výky na gymnáziích by měl být základ 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 oběhuje 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 cíly 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  *užijeme reprezentovat mnoha způsoby. Ve vzdělávacím obsahu se pak oběhuje kategorie***2.???***Je to kategorie, nebo se to nazývá jinak? Zpracování a prezentace inforací,* kde jedním z očekávaných výstupu je, že žák *aplikuje algoritmický přístup k řešení problému.* 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.**3.lépe**

Vzdělávání na odborných středních školách probíhá samozřejmě podle RVP také, každý obor má svůj vlastní dokument. Analyzujeme nyní výskyt algoritmizace a programování na RVP oboru Informační technologie **4.citace**, kde by úroveň informatického vzdělávání měla být nejvyšší z celého středoškolského systému. Programování najdeme v tzv. klíčových odborných kompetencích, konkrétně klíčová kopetence 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.

Pro odborné školy je tedy vzdělávací obsah blíže specifikován. 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, taže 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 (skrže dětské programovací jazyky) na prvním stupni, některé vůbec programování nenasazují.³ Obecně lze říci, že výuka ICT je směřována spíše

³Měl 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.

Tabulka 2.1: Obsahový okruh *programování a vývoj aplikací*

Výsledky vzdělávání	Učivo
Žák: <ul style="list-style-type: none"> • zná vlastnosti algoritmu; • zanalyzuje úlohu a algoritmizuje ji; • zapíše algoritmus vhodným způsobem; 	1 Algoritmizace <ul style="list-style-type: none"> • význam, prvky algoritmu
<ul style="list-style-type: none"> • použije základní datové typy; • použije řídicí struktury programu; • vytvoří jednoduché strukturované programy; 	2 Strukturované programování <ul style="list-style-type: none"> • datové typy • řídicí struktury
<ul style="list-style-type: none"> • rozumí pojmům třída, objekt a zná jejich základní vlastnosti; • použije jednoduché objekty; 	3 Úvod do objektového programování <ul style="list-style-type: none"> • třída, objekt, vlastnosti tříd
<ul style="list-style-type: none"> • zná výhody použití jazyka SQL; • použije základní příkazy jazyka SQL; 	4 Základy jazyka SQL <ul style="list-style-type: none"> • základní příkazy (SELECT, UPDATE, INSERT, DELETE)
<ul style="list-style-type: none"> • 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



k ovládání informačních a komunikačních technologií, než k práci s algoritmizací a programováním. Pokud bychom měli tvořit vysokoškolskou přípravu podle RVP, v programu **Z** by se programování nemuselo objevit vůbec, stačilo by osvojení algoritmizace. V programu pro SŠ už se zcela jistě programování objevit musí, nejsme ale omezeni konkrétním programovacím jazykem. Pregraduální příprava z pohledu RVP by také měla obsahovat jak průpravu do paradigmatu strukturovaného programování, tak i objektově orientovaného programování, tak aby byl absolvent schopen zajistit výuku na střední odborné škole oboru informatika.

2.4.1 Deklarované kurikulum výuky informatiky na Slovensku

Jelikož je informatika mladý obor a změny v něm opravdu rychlé, kurikulumní dokumenty nemusí obsahovat nejaktuálnější trendy v oboru. Analyzujeme proto zakotvení programování ve slovenských kurikulumních dokumentech. Štátný vzdělávací program je tomu českému velmi podobný, vznikl ale o trochu později. Porovnání těchto kurikulumních dokumentů pro úroveň ISCED 1 a 2 provedl Berki (?), nedostatečná specifikace vzdělávacího obsahu a absence práce s informatikou jako vědou a algoritmickým myšlením byly jeho hlavní zjištění. (? , 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

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

5.citace

Výkonový štandard	Obsahový štandard
<ul style="list-style-type: none"> • 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 použitia 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. 	<ul style="list-style-type: none"> • Problém. Algoritmus. Algoritmy z bežného 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.

- Informačná spoločnosť

Algoritmizaci se věnuje okruh *Postupy, riešenie problémov, algoritmické myslenie*, ve kterém jsou dále definovány obsahové a výkonové standardy⁴. 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. (? , s. 85)

Slovensko ale v inovaci pokračovalo a v roce 2015 nasadilo Inovovaný štatný vzdělávací program, který všechny oblasti ještě dále specifikuje. Uvedme nyní jako příklad opět tabulku s obsahovými a výkonovými standarty předmětu informatika pro úroveň ISCED3 okruhu *Postupy, riešenie problémov, algoritmické myslenie* pro 5.–8. ročník víceletého gymnázia⁵:

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

⁵Na 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*

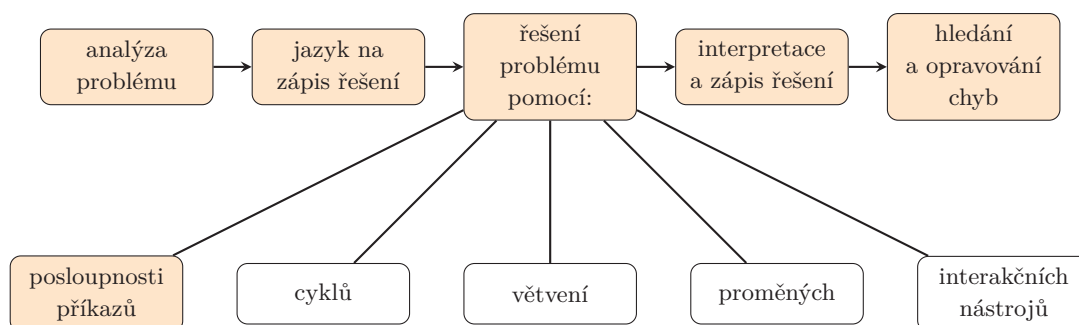
Výkonový štandard	Obsahový štandard
Analýza problému	
<p>Žiak vie/dokáže</p> <ul style="list-style-type: none"> identifikovať vstupné informácie zo zadania úlohy, popísať očakávané výstupy, výsledky, akcie, identifikovať problém, ktorý sa bude riešiť algoritmicky, formulovať a neformálne (prirodzeným jazykom) vyjadriť ideu riešenia, uvažovať o vlastnostiach vykonávateľa (napr. korytnačka, grafické pero, robot, a pod.), naplánovať riešenie úlohy ako postupnosť príkazov vetvenia a opakovania 	<p>Vlastnosti a vzťahy: zadaný problém – vstup – výstup</p> <p>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</p> <ul style="list-style-type: none"> význam, prvky algoritmu
Jazyk na zápis riešenia	
<ul style="list-style-type: none"> 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. 	<p>Pojem: program, programovací jazyk</p> <p>Vlastnosti a vzťahy: zápis algoritmu a vykonanie programu, vstup – vykonanie programu – výstup/akcia</p> <p>Procesy: zostavenie programu, identifikovanie, hľadanie, opravovanie chýb</p>
Pomocou postupnosti príkazov	
<ul style="list-style-type: none"> riešiť problém skladaním príkazov do postupnosti, aplikovať pravidlá, konštrukcie jazyka pre zostavenie postupnosti príkazov. 	<p>Pojmy: príkaz, parameter príkazu, postupnosť príkazov</p> <p>Vlastnosti a vzťahy: ako súvisia príkazy a výsledok realizácie programu</p> <p>Procesy: zostavenie a úprava príkazov, vyhodnotenie postupnosti príkazov, úprava sekvencie príkazov (pridanie, odstránenie príkazu, zmena poradia príkazov)</p>
Pomocou nástrojov na interakciu	
<ul style="list-style-type: none"> rozpoznávať situácie, kedy treba získať vstup, identifikovať vlastnosti vstupnej informácie (obmedzenia, rozsah, formát), rozpoznávať situácie, kedy treba zobrazíť výstup, realizovať akciu, zapisovať algoritmus, ktorý reaguje na vstup, vytvárať hypotézu, ako neznámy algoritmus spracováva zadaný vstup, ak sú dané páry vstup–výstup/akcia. 	<p>Vlastnosti a vzťahy: prostriedky jazyka pre získanie vstupu, spracovanie vstupu a zobrazenie výstupu</p> <p>Procesy: čakanie na neznámy vstup – vykonanie akcie – výstup, následný efekt</p>

Pomocou premenných	
<ul style="list-style-type: none"> 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 	<p>Pojmy: premenná, meno (pomenovanie) premennej, hodnota premennej, operácia (+, -, *, /)</p> <p>Vlastnosti a vzťahy: pravidlá jazyka pre použitie premennej, meno premennej – hodnota premennej</p> <p>Procesy: nastavenie hodnoty (priradenie), zistenie hodnoty (použitie premennej), zmena hodnoty premennej, vyhodnocovanie výrazu s premennými, číslami a operáciami</p>
Pomocou cyklov	
<ul style="list-style-type: none"> 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. 	<p>Pojmy: opakovanie, počet opakovaní, podmienka vykonávania cyklu, telo cyklu</p> <p>Vlastnosti a vzťahy: ako súvisí počet opakovaní s výsledkom, čo platí po skončení cyklu</p> <p>Procesy: vyhodnotenie hraníc/podmienky cyklu, vykonávanie cyklu</p>
Pomocou vetvenia	
<ul style="list-style-type: none"> 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. 	<p>Pojmy: vetvenie, podmienka</p> <p>Vlastnosti a vzťahy: pravda/nepravda – splnená/nespĺnená podmienka</p> <p>Procesy: zostavovanie a upravovanie vetvenia, vytvorenie podmienky a vyhodnotenie podmienky s negáciami a logickými spojkami (a, alebo)</p>
Interpretácia zápisu riešenia	
<ul style="list-style-type: none"> krokovat riešenie, simulovať činnosť vykonávateľa s postupnosťou 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, doplňať, dokončovať, modifikovať rozpracované riešenie, hľadať vzťah medzi vstupom, algoritmom a výsledkom, uvažovať o rôznych riešeniach, navrhovať vylepšenie. 	<p>Vlastnosti a vzťahy: jazyk - vykonanie programu</p> <p>Procesy: krokovanie, čo sa deje v počítači v prípade chyby v programe</p>

Hľadanie a opravovanie chýb	
<ul style="list-style-type: none"> 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. 	<p>Vlastnosti a vzťahy: chyba v postupnosti príkazov (zlý príkaz, chýbajúci 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</p>



Výhodou je, že toto dělení je zachováno pro všechny stupně vzdělávání, snadno se identifikuje posun žáků v jejich znalostech a dovednostech jak píše Berki (? , 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 (obrázek 2.3), 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 součástí výuky informatiky na základních a středních školách.



2.5 Proč učit programování?



6. Chybí citace, je potřeba trošku uhladit Programování je běžně součástí pregraduální přípravy učitelů, ačkoliv se neseťkává u studentů s velkou oblibou. Studenti mají obtíže předměty zaměřené na programování absolvovat, „a proto by se raději ve své budoucí pedagogické praxi výuce programování vyhnuli.“ Studenti se staví spíše proti výuce programování na základních školách, má být podle nich součástí 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 nezůstává 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 amělo být programování a algoritmizace součástí kurikula jak na vysoké škole tak na nižších stupních. Výuka programování a algoritmizace není příprava na povolání programátora, ale „Cílem programování ve škole je rozvoj tvořivosti a myšlení. Samotné programování je (skvělým) nástrojem k dosahování těchto cílů.“ (<http://ucime-informatiku.blogspot.cz/2015/10/povinne-programovani-od-prvni-tridy.html>) 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 Schwilllová 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.“ (didaktika na startu). Na Slovensku se setkávají žáci s informatickými tématy jako algoritmické myšlení procedury a princip fungování digitálních technologií už od 3. třídy. V Anglii je zaveden je od září 2014 se vyučuje předmět Computing, (cas, didaktika inf. Na startu) jehož cílem je rozumět a aplikovat základní principy informatiky. 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 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 nadčasové⁶. Podle Paperta někteří žáci mají model učení postavený 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 programování prochází v poslední době změnami, kteří se snaží reagovat dynamický rozvoj softwarového průmyslu. Cílem výuky programování není jen schopnost umění programovat, ale také schopnost vytvářet algoritmy a logicky přemýšlet. 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 výzce 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”. Výuka může probíhat i pomocí jiných metodik, představme si krátce některé z nich:

Object first Filozofie této metodiky se snaží studentovi v první řadě představit důležité koncepty objektově orientovaného programování [7.citace - stránky blueJ](#), aniž by se musel zabývat syntaxí programovacího jazyka. Autoři 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.



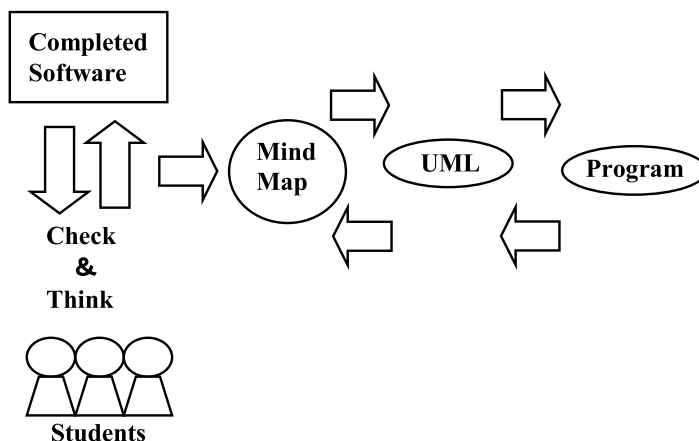
Architecture first Tato metodika se snaží rozšiřovat pojetí Object first, ke které přidává ještě větší důslednost na znalost architektury softwaru, než se přejde k samotnému kódování programu.

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.

Agilní metodiky Tento pojem zastřešuje přístup k vývoji softwaru pomocí několika principů např. "Nejúčinnějším a nejefektivnějším způsobem sdělování informací vývojovému týmu z vnějšku i uvnitř něj je osobní konverzace." nebo "Hlavním měřítkem pokroku je fungující software". Studenti v tomto přístupu

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

zastávají 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 (?, s. 56)

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

Důležitým aspektem ve výuce programování je nejen 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 ...

Uvedme si nyní několik programovacích jazyků, které jsou v současnosti relevantní ve výuce programování. **Jak jsem tyto jazyky volil**

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 **9.**<http://cacm.acm.org/blogs/blog-cacm/176450-python-is-now-the-most-popular-introductory-teaching-language-at-top-u-s-universities/fulltext>, může být použit i kvýuce na základních školách. "Jazyk sa rýchlo a dobre učí. Programovanie je pragmatické a rýchlo vedie k stručnému a efektívnemu programovému kódu." **10.**Skúsenosti s programovaním na základnej škole,didinfol2015

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++. **11.**Java Programming Language ,Kamran Sartipi,August, 1996 Hojně se využívá i k výuce programování na vysokých školách. Její výhodou je značná nezávislost na zařízení, je tak uživna např. na mobilních zařízeních jako mobilní telefony nebo tablety. Pro Javu vzniklo několik vývojových prostředí vhodných pro výuku programování, které umožňuj vizualizovatí jednotlivé třídy a objekty programu jako např BlueJ nebo Greenfoot.

Scratch Programovací jazyk Scratch je speciálně vytvořen pro výuku dětí. Patří mezi vizuální programovací jazyky, programování tak probíha 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.**12.** <http://jtie.upol.cz/pdfs/jti/2013/01/15.pdf>


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". 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."**13.**<http://bit.ly/2IKDFKC>

Jak píše Lessner 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.

3 Analýza deklarovaného kurikula pregraduální přípravy

Příprava budoucích učitelů informatiky probíhá v ČR na přírodovědných 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 zde 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 vzniknul vzorek 25 studijních programů ze 17 fakult 10 univerzit. 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 kategorií byl předmět zařazen v případě, pokud se v něm vyskytuje níže popsany výraz v jakékoliv míře. Tímto způsobem se dá

- **Výuka programování** – do této kategorie byly zařazeny všechny předměty, jejímž obsahem je výuka programování či teorie přímo související s programováním 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, jejichž součástí je programování v jazyku JavaScript nebo PHP.
- **Didaktické předměty** – do této kategorie je zařazen předmět, pokud alespoň částí jeho obsahu je didaktika programování nebo didaktika algoritmizace.

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

4 Textová analýza

4.1 Popis vzorku

4.2 Popis způsobu analýzy

4.3 Výsledky



5 Návrh konceptu pregraduální přípravy

5.1 ZŠ

5.2 SŠ

6 Závěr

Lorem ipsum dolor sit amet, consectetur 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, consectetur 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, consectetur 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, consectetur 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, consectetur 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, consectetur 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, consectetur 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, consectetur 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, consectetur 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, consectetur 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.

Seznam tabulek

2.1	Obsahový okruh <i>programování a vývoj aplikací</i>	16
2.2	Obsahový okruh <i>Postupy, riešenie problémov, algoritmické myslenie</i> z ŠVP 2011	17
2.3	Obsahový okruh <i>programování a vývoj aplikací</i>	18