# MODERN APPROACHES TO TEACHING PROGRAMMING

*Rostislav FOJTÍK*

***Abstract:*** The article bases from practical experience of the author teaching programming on secondary and tertiary level of education. The aim is to show suitable methods in teaching of object-oriented programming. The results of the surveys in primary and secondary schools, we can say that the teaching of programming and algorithms are not sufficiently exploited. The author tries to find most frequent problems of this teaching and at the same time its subjects matter and methods in generally use and present some suggestions of possible solution.

***Key words:*** object-first, object-oriented programming, teaching programming.

## MODERNÍ PŘÍSTUPY K VÝUCE PROGRAMOVÁNÍ

***Resumé:*** Cílem příspěvku je představit vhodné moderní způsoby výuky programování. Podle výsledků provedených průzkumů na základních a středních školách je možné konstatovat, že výuka programování a algoritmizace není dostatečně využívaná. Případně se používají méně vhodné postupy a metodiky pro výuku programování. Na základě provedených experimentů ukazuje příspěvek možnosti výuky programování metodikou object-first, její výhody i úskalí. Výběr příhodných postupů při výuce a vhodných vývojových nástrojů.

***Klíčová slova:*** object-first, objektově orientované programování, výuka programování.

## 1 Úvod

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.

Učitelé se mnohdy ptají, který programovací jazyk je vhodný pro výuku programování. První otázka při vytváření kurzu programování by se však neměla týkat programovacího jazyka, ale výběru vhodného paradigmatu a cíle kurzu. Je-li cílem naučit studenty vytvářet algoritmy a logicky přemýšlet, jeví se jako nejvhodnější mikrosvěty, například Karel nebo Logo. Tyto nástroje jsou přehledné, jednoduché a velmi názorné. Studenti vidí pomocí grafického znázornění výsledky svého postupu. Nástroje nevyžadují dlouhou fázi učení pravidel jazyka a uživatele se mohou skutečně zaměřit na tvorbu algoritmů a jednoduchých programů. Neztrácejí se ve složitostech a konkrétních detailech jazyka.

Je-li však cílem výukového kurzu naučit studenty programovat a to tak, aby se jejich znalosti a dovednosti daly využít následně i v praxi, je potřeba nejprve vybrat vhodné paradigma. V současné době je nejrozšířenějším objektově orientovaný přístup. Nové přístupy v oblasti programování však vyžadují rovněž nové metodické postupy ve výuce. [7]

## 2 Jak se v současnosti vyučuje programování?

Rámcový vzdělávací program pro gymnázia obsahuje jen velmi obecně popsané kompetence v oblasti algoritmizace a programování. V dokumentu se nacházejí pouze pojmy algoritmus, zápis algoritmů, úvod do programování. [8]

Podíváme-li se do požadavků potřebných pro složení společné části maturitní zkoušky v oblasti informatiky, zjistíme, že v základní úrovní se od žáků očekává jen schopnost vysvětlení pojmu algoritmus, jeho základní vlastnosti a algoritmizovat jednoduchou úlohu. Nejedná se tedy o znalosti a dovednosti v oblasti programování, ale pouhé algoritmizace. V takovém případě není nutné vyučovat programování pomocí programovacích jazyků, ale jde využít například mikrosvěty. U vyšší úrovně maturitní zkoušky již musí žák prokázat znalosti a dovednosti skutečného programování a jedním z požadavků je i znalost základů objektově orientovaného programování. [4][5]

Výuka programování na mnoha středních školách však neprobíhá podle současných požadavků. Velká část pedagogů vyučuje stejnými nebo velmi podobnými přístupy jako byli učeni sami v době svého studia. Problém je, že současné nároky na vývoj programů se značně změnily. Dříve nejběžnější procedurální

paradigma a přístup imperative-first nebo algorithm-first mnohdy nedostačují. Zvláště je to vidět u v současnosti velmi využívaného objektově orientovaného programování (OOP). Mnozí vyučující uplatňují nejprve klasicky procedurální přístup a teprve následně se začínají věnovat principům objektově orientovanému programování. Tento přístup však u mnohých studentů vytváří nevhodné návyky, které se obtížně odstraňují. Z praktických zkušeností na Katedře informatiky a počítačů Ostravské univerzity vyplývají následující poznatky. Studenty přicházející ze středních škol můžeme rozdělit na tři skupiny. První skupinou jsou studenti, kteří jsou výukou programování nedotčeni. Druhá skupina pochází ze škol, kde se programování věnovali, ale vzhledem k malé hodinové dotaci jen v omezené míře. Do poslední skupiny patří studenti, ze škol, které věnují programování samostatný předmět. V dobách, kdy se v předmětu věnovanému základům programování využíval programovací jazyk Pascal a imperativní přístup, se mezi studenty ve velké míře objevovaly dále popsané chyby a špatné návyky. Pro první skupinu studentů bylo typické, že se potýkala hlavně s programovacím jazykem a jeho pravidly. Většinu času věnovali studenti zvládnutí syntaxe a na samotné algoritmy jim nezbýval dostatek energie. Pro druhou skupinu studentů bylo naopak mnohdy charakteristické velké množství nevhodných návyků. Mnoho studentů z této skupiny programování považovalo za velmi složité a mělo pocit, že jej nemohou nikdy pochopit. Studenti vycházeli obvykle z dosavadních nedobrých zkušeností, které byly způsobeny špatným výběrem programovacího jazyka, vývojového nástroje, nekvalitními a málo názornými příklady a nedostatkem času na výuku. Další problémy nastaly v navazujících předmětech, které byly zaměřeny na objektově orientované programování. Studenti museli zvládnout nový programovací jazyk a hlavně nové paradigma, které od nich vyžadovalo zásadně jiný přístup v návrhu programů. Situace se výrazně změnila v okamžiku, kdy úvodní kurz zaměřený na základy programování se začal vyučovat pomocí programovacího jazyka Java a byl doplněn o objektově orientované paradigma.

Dosavadní zkušenosti s výukou programování ukazují, že mezi obvyklé chyby studentů, kteří se učí programování nejprve pomocí klasického přístupu a teprve pak objektového, patří:

- přemýšlí hlavně o kódu – jazyku, ne o řešení problému,
- studenti mají tendenci vytvářet monolitická řešení, ve kterém spojují funkční logiku, strukturu a uložení dat s uživatelským rozhraním,
- nedostatečné využívání návrhových vzorů,
- nevhodné používání globálních proměnných uvnitř podprogramů,
- již na počátku se snaží řešit nejdetailnější problémy,
- vytvářejí velké třídy s moha kompetencemi,
- třídy obsahují nevhodné kompetence.

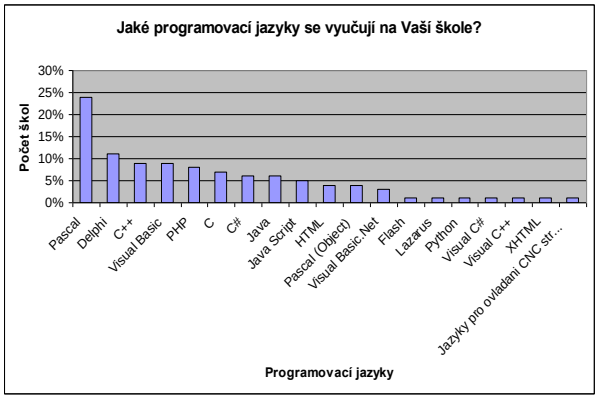
Podíváme-li se na tematické plány předmětů programování na středních školách, najdeme dost často následující schéma. Začátek je věnován algoritmizaci, pak následuje popis programovacího jazyka. Jednotlivá témata vysvětlují syntaktická pravidla jazyka. Žáci se dozví o jednotlivých datových typech, vytváření proměnných, operátorech, co jsou to cykly, podmínky, pole a podobně. Na konci (pokud na to vyjde čas) se věnuje výuka základům objektového programování. Dozví se, co je třída, objekt, dědičnost, polymorfismus. Žáci z tohoto postupu získají pocit, že objektové programování je jen určitá nadstavba jazyka. Jedním z důvodů tohoto postupu výuku je rovněž zaměření učebnic. Podíváme-li se na nabídku knihkupectví v sekci programování, zjistíme, že velká část knih je zaměřena na programování v konkrétním jazyce. V úvodních kapitolách se autoři věnují datovým typům, od těch primitivních až po ty složitější a strukturované, dále jsou na mnoha stránkách popisovány základní řídící struktury, práce s poli, přístup k souborům, vytváření programových modulů a podobně. Popis všech pravidel konkrétního jazyka zabírá mnoho desítek až stovek stránek. O analýze a návrhu programu, návrhových vzorech a vhodných programovacích postupech se kniha zmiňuje často jen velmi okrajově nebo vůbec ne. Nezbývá na ně místo! Místo učebnic programování se pak tisknou spíše referenční manuály pro konkrétní programovací jazyk. Taková učebnice je vhodná pro člověka, který umí programovat a potřebuje zvládnout nový programovací jazyk. Ke škodě věci se pak velmi často podobným způsobem vyučuje programování a učitele si knihu berou jako vzor postupu výuky. Hlavní důraz je kladen na zvládnutí syntaktických pravidel a úskalí vybraného programovacího jazyka.

Než učitel sestaví tematický plán, měl by si ujasnit cíl výuky programování:

- Naučit základům algoritmizace – zde je vhodné využívat jednoduché a názorné prostředky, jako jsou například mikrosvěty.
- Zvládnout konkrétní jazyk – žáci už znají základy programování a vzhledem k profilu absolventa daného studijního oboru mají zvládnout konkrétní programovací jazyk.
- Naučit se programovat podle moderních měřítek – to znamená zaměřit se na objektově orientovaný přístup, který je uplatňován v současné praxi nejčastěji.

## 3 Průzkum na základních a středních školách

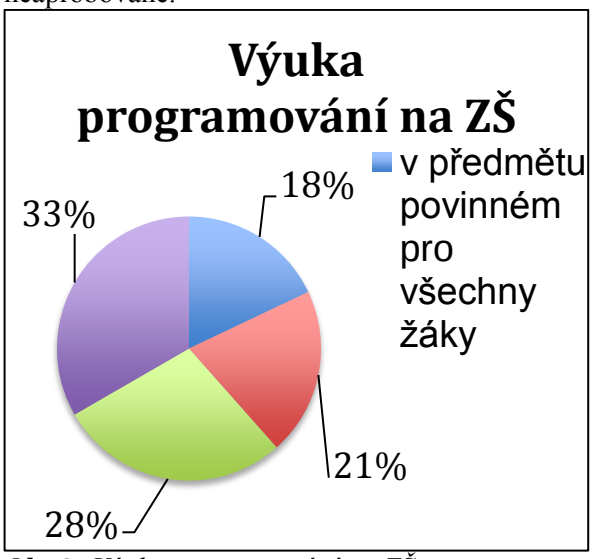
Hlavním cílem průzkumu bylo zjistit, jakým způsobem přistupují základní a střední školy k výuce programování. Zda zařazují do svých tematických plánů výuku algoritmizace a programovaní, jaká vývojová prostředí a jaké metodiky využívají ve výuce. [2][3] První samostatně zkoumanou skupinou bylo 63 středních škol v Moravskoslezském kraji, z nichž 84% uvedlo, že se věnuje výuce programování. Nejčastěji využívaným jazykem byl Pascal. Na odpovědích je bohužel vidět, že někteří učitelé si pletou vývojové prostředí s programovacím jazykem. Příkladem může být vývojový nástroj Delphi, který přes 10 % respondentů mylně považuje za programovací jazyk.




***Obr 1:*** *Využití programovacích jazyků na koumaných středních školách.*

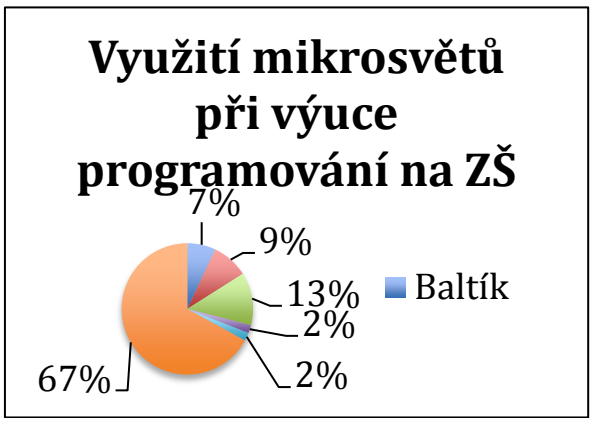
Jeden z hlavních důvodů využívání jazyka Pascal (případně novější varianty Object Pascal) je to, že se učitelé sami pomoci něj učili programovat. Je otázkou nakolik je jazyk Pascal vhodný pro moderní výuku programování. Je potřeba si uvědomit, že přestože Niklaus Wirth koncipoval začátkem 70.let minulého století jako jazyk pro výuku, byl to jazyk poplatný hlavně tehdejším požadavkům. Nezahrnoval objektově orientovaný přístup, který byl zaveden do jazyka až mnohem později, a to mnohdy ne úplně vhodným způsobem.

Druhou zkoumanou skupinou bylo 71 základních škol a 15 víceletých gymnáziích (nižší stupeň). Z tohoto počtu jen 32 % respondentů zařadilo do svých tematických plánů v rámci předmětu výpočetní technika nebo v zájmovém kroužku výuku programování. 22 % respondentů vyučovalo programování v dřívějších dobách a zbylých 46 % programování nevyučovalo ani nevyučuje. Jedním z důvodů malého zastoupení výuky programování je i nadále nedostatečná aprobace vyučujících. Stále jen malá část učitelů má informatiku jako svou původní aprobaci (pouze 15 % respondentů). Více učitelů informatiku absolvovalo v rámci některé z forem rozšiřujícího studia (30 % respondentů) nebo učí informatiku neaprobovaně.




***Obr 2:*** *Výuka programování na ZŠ*

Pro výuku základů algoritmizace a programování používá 33 % dotazovaných základních škol *mikrosvěty*. Jedná se o jednoduché a většinou uzavřené vývojové nástroje, jejichž hlavní výhodou je jednoduché ovládání a hlavně velká názornost. Tyto nástroje jsou mnohdy uzpůsobeny pro menší děti, ale dají se úspěšně aplikovat i na střední škole. Nejčastěji jsou používané Logo, Karel a Baltík.




***Obr 3:*** *Využití mikrosvětů ve výuce programování na ZŠ.*

## 4 Experiment

Katedra informatiky a počítačů Ostravské univerzity se snaží pravidelně pořádat kurzy výuky programování pro zájemce z řad studentů středních škol. Hlavním cílem je seznámit účastníky s moderními vlastnostmi programování s důrazem na objektově orientovaný přístupu.

V rámci výuky byl proveden pedagogický experiment. Pro studenty byly připraveny dva kurzy v rozsahu 20 hodin prezenční výuky a materiály pro samostudium. Každého kurzu se zúčastnilo od 12 do 28 středoškolských studentů. Jejich znalosti programování byly na různé úrovni. Třetina studentů se zatím neučila žádný programovací jazyk. Zbylá část studentů uměla základy jazyka C nebo Pascal. Alespoň základy objektově orientovaného programování znalo jen několik studentů. Jediný náznak objektově orientovaného přístupu měli studenti, kteří při výuce na své škole využívali vývojový nástroj Delphi. Ale i ti o třídách a objektech věděli jen velmi málo. Hlavní důraz jejich výuky byl zaměřen spíše na využití grafických komponent a návrh grafického rozhraní aplikace. Při svých programech většinou nevyužívali architekturu Model-View-Controler.

V kontrolní skupině byla výuka vedena tradičními postupy. Jednotlivá setkání byla nejprve věnována vlastnostem programovacího jazyka Java, syntaktickým pravidlům, jednoduchým a strukturovaným datovým typům, řídícím strukturám jazyka a podobně. Teprve v závěru se výuka věnovala základům objektově orientovaného jazyka. Experimentální skupina byla po nezbytném úvodu hned seznamována s paradigmatem objektově orientovaného programování. Metodika výuky vycházela z přístupu Object-First. [1][6] Syntaktická pravidla se „dovysvětlovala za pochodu“ a nevěnovala se jim hlavní pozornost. Vzhledem k omezenému času to nebylo ani možné. Hlavní důraz při výuce byl kladen na pochopení objektově orientovaného přístupu, konstrukci tříd, vztahů mezi objekty, využití návrhových vzorů a tvorba správné architektury.

Při rozhodování o výběru programovacího jazyka byl vybrán pro kontrolní i experimentální skupinu programovací jazyk Java. Jeho vlastnosti v současné době vyhovují výuce:

- umožňuje objektově orientovaný přístup,
- možnost využít jazyk přímo v praxi je pro studenty motivující. Většina studentů nerada pracuje s programovacími jazyky, které jsou určeny pouze k výuce či teoretickému bádání,
- jazyk je jednodušší než C++,
- syntaxe jazyka je podobná jako u jazyků C, C++, C# a některých dalších často využívaným programovacích a skriptovacích jazyků,
- ke grafickému znázorňování tříd a jejich vztahů lze využit pro výuku vhodné prostředí BlueJ,
- je k dispozici zdarma kvalitní vývojov0 prostředí (v kurzech byly využit nástroj NetBeans).

Výuka v experimentální skupině byla prováděna hlavně pomocí řešení konkrétních problémů a programů. Cílem nebylo vytvářet komplexní a dokonalé programy, ale ukázat možné objektově orientované postupy. A to mnohdy i za cenu velkého zjednodušení problému a následného řešení. Vytvářet složité a rozsáhlé kódy by nebylo vhodné a jen velmi obtížně by se realizovalo ve výuce s takto nevelkou časovou dotací. Naopak veliký důraz byl kladen na diskusi a komunikaci mezi všemi účastníky výuky. Při řešení jednotlivých úkolů bylo nejprve využito grafického znázornění problému a teprve pak následovalo psaní samotného kódu. Grafické znázornění je velmi důležité a usnadňuje studentům pochopení řešení.

Velmi důležitou roli hrály vhodně zvolené příklady a problémové úkoly. Jejich příprava a hlavně vymýšlení je velice obtížné. Příklady, které studenti během výuky řeší, musí často splňovat téměř protichůdné požadavky:

- musí být dostatečně komplexní, aby dostatečně pokryly řešenou oblast,
- měly by sloužit jako vzor pro řešení reálných problémů,
- měly by mít alespoň částečnou spojitost s reálným využitím, neměly by to být pouze „příklady pro příklady“,

- jejich řešení nesmí být rozsáhlé a komplikované natolik, že se v čase vyučovací hodiny nedají vyřešit.

Přestože účast studentů na seminářích nebyla povinná, byla docházka většiny z nich překvapivě dobrá.

K vyhodnocení byly použity metody pozorování a interview se studenty. Při srovnání obou skupin bylo pomocí pozorování zjištěno, že účastníci kontrolní skupiny lépe zvládli syntaktická pravidla programovacího jazyka Java a některé algoritmické konstrukce. Vzhledem k omezené časové dotaci, však ani znalosti účastníků kontrolní skupiny nebyly dostatečné k tomu, aby žáci mohli okamžitě tvořit reálné projekty. To by vyžadovalo další hodiny výuky a samostudia. V kontrolní skupině bylo možné pozorovat velké rozdíly mezi studenty, kteří se na své střední škole programování věnovali a studenty, kteří doposud, žádný programovací jazyk neznali. Všichni studenti kontrolní skupiny však oproti členům experimentální skupiny nedostatečně pochopili základy objektově orientovaného přístupu, činilo jim větší potíže definovat třídy, objekty, vztahy mezi nimi a hlavně mnohem hůře rozuměli praktickému využití OOP. V experimentální skupině se často ukazovalo, že předchozí znalosti programování nemusí být výhodou. Naopak občas studenti měli problém změnit způsob uvažování.

## 5 Závěr

Současné požadavky na programátorské dovednosti se postupně mění. Programátoři v praxi stále více potřebují správně chápat objektově orientovaný přístup v oblasti návrhu a tvorby programů. Proto již nestačí vyučovat programování s hlavním důrazem na algoritmizaci a zvládnutí programovacího jazyka. Praktické zkušenosti z výuky ukazují, že metodika Object-First je vhodnější pro získání správných návyků a kvalitnější pochopení problematiky objektově orientovaného programování. Pro studenty je důležité správně porozumět analýze, návrhu a architektuře programů.

## 6 Literatura

[1] BENNEDSEN, J., SCHULTE, C. What does "Objects-First" Mean?, *Proc. Seventh Baltic Sea Conference on Computing Education Research,* Finland 2007, [online] http://crpit.com/confpapers/CRPITV88Bennedsen.pdf

[2] FOJTÍK, R., JIRÁSKOVÁ, D. Programování na základních a středních školách. *Objekty 2009*. Gaudeamus, Univerzita Hradec Králové, 2009. s. 75-83. ISBN 978-80-7435-009-2

[3] FOJTÍK, R., DROZDOVÁ, M. Teaching of programming at secondary schools. *ICTE 2009.* Ostrava: University of Ostrava, 2009. s. 77-81. ISBN 978-80-7368-459-4

[4] *Katalog požadavků zkoušek společné části maturitní zkoušky, Zkušební předmět: Informatika, vyšší úroveň obtížnosti,* Centrum pro zjišťování výsledků vzdělávání, Praha 2010

[5] *Katalog požadavků zkoušek společné části maturitní zkoušky, Zkušební předmět: Informatika, základní úroveň obtížnosti,* Centrum pro zjišťování výsledků vzdělávání, Praha 2010

[6] PECINOVSKÝ, R. Jak efektivně učit OOP. *Tvorba softwaru 2005*, ISBN 80-86840-14-X.

[7] PECINOVSKÝ, R. Výuka programování podle metodiky Design Patterns First**,** *Tvorba software 2006*, ISBN 80-248-1082-4

[8] *Rámcový vzdělávací program pro gymnázia,* Výzkumný ústav pedagogický, Praha 2007, ISBN 978-80-87000-11-3

**Mr. Rostislav Fojtík, PhD.**
**Katedra informatiky a počítačů**
**Přírodovědecká fakulta UO**
**30.dubna 22**
**701 00, Ostrava, ČR**
**Tel: +420 603 167 768**
**E-mail: rostislav.fojtik@osu.cz**
**Www pracoviště: http://prf.osu.cz/kip/**