# PODPORA VÝUKY ALGORITMIZACE IT NÁSTROJI

JELÍNEK Jiří, CZ

**Resumé**

Úroveň schopností studentů algoritmizovat nejrůznější typy úloh se neustále zhoršuje. Je možné diskutovat o rozsahu výuky tohoto tématu na různých typech a stupních škol. Z pohledu pedagoga je však podstatnější zajištění a nastavení procesu podpory výuky nástroji, které by bylo možné využít pro zvýšení její celkové efektivity. Na toto téma se soustřeďuje předložený příspěvek, ve kterém po úvodu do problematiky identifikujeme hlavní kategorie podpůrných nástrojů podle způsobu prezentace algoritmů a tyto kategorie se pokusíme následně porovnat z hlediska jejich užití pro výuku algoritmizace u cílové skupiny začátečníků bez dalších znalostí použitelných v daném tématu. Prezentovány budou i některé konkrétní nástroje zastupující jednotlivé kategorie a také výsledky porovnání.

**Klíčová slova:** podpora výuky, algoritmizace, algoritmy, informační technologie.

## SUPPORT OF TEACHING ALGORITHMIZATION WITH IT TOOLS

**Abstract**

The level of students' abilities to create algorithms for all sorts of tasks continues to decrease. We may discuss the extent of teaching this topic in different types and levels of schools. However, from the perspective of a teacher is more important to ensure and set support of teaching with tools that could be used to improve its overall efficiency. On this topic is focused also this paper, in which, after introduction into the problem, we identify major categories of support tools defined by the presentation of algorithms. These categories we will then try to compare from the point of view of their use for teaching algorithms for the target group of beginners without other special knowledge applicable in a given topic. Some specific instruments representing the different categories and the results of comparison will be presented too.

**Key words:** education support, algorithmization, algorithms, information technology.

## Úvod

Úroveň schopností studentů algoritmizovat nejrůznější typy úloh se neustále zhoršuje. Je možné diskutovat o rozsahu výuky tohoto tématu na různých typech a stupních škol. Z pohledu pedagoga je však podstatnější zajištění a nastavení procesu podpory výuky nástroji, které by bylo možné využít pro zvýšení její celkové efektivity.

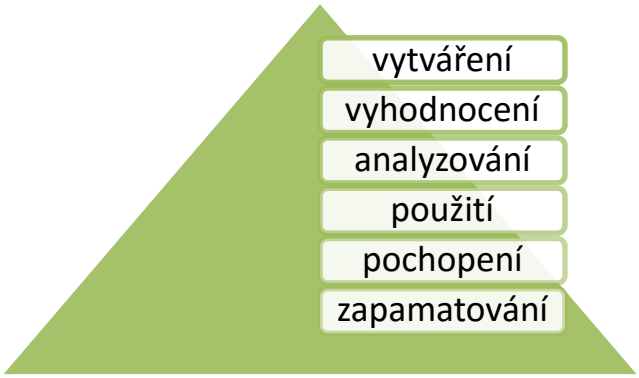
Algoritmizace je často spojována pouze s oblastí programování. Faktem však je, že schopnost definovat postupy k dosažení stanovených cílů má výrazně širší oblast užití a jedinci jí vybavení mají výrazně lepší uplatnitelnost při samostatném řešení zadaných úkolů. Úzký vztah k programování je dán především stejnými základními předpoklady a schopnostmi, mezi které patří systémový přístup k problému či přesnost ve vyjadřování i popisu jednotlivých kroků.

Ačkoliv existuje celá řada kategorií algoritmů, v našem textu se zaměříme na klasické pojetí s přímým možným uplatněním v kurzech základů programování či dalších oblastech vyžadujících popis řešení zvolených problémů.

## 1 Současný stav

Na proces algoritmizace je možné nahlížet z pohledu jedince, jeho chování a kognitivních procesů. Klíčové jsou zde zejména cíle jedince, přičemž algoritmy může využívat k návrhu postupů

pro jejich dosažení. Při tvorbě algoritmu jedinec využívá své znalosti a zkušenosti získané interakcí se svým okolím. Na nich pak staví svůj vlastní model okolního světa, jehož kvalitu (hloubku pochopení problému) můžeme popsat pomocí Bloomovy taxonomie cílů učení v její aktualizované podobě (1). Je zřejmé, že pro úspěšnou tvorbu algoritmů je nutné pochopení problému na nejvyšší úrovni, umožňující úspěšnou aplikaci dílčích získaných znalostí a jejich integraci do nového celku – algoritmu.




*Obr. 1 Aktualizovaná Bloomova hierarchie učení (chápání) (zdroj: vlastní)*

Jak z předchozího vyplývá, algoritmizace je problémově, či spíše doménově závislou činností, neboť v konkrétní doméně mohou být využívány různé elementární komponenty, ze kterých lze algoritmus sestavit. Znalosti potřebné pro tvorbu algoritmu se tak dají rozdělit do dvou základních oblastí:

- Obecné znalosti o základních algoritmických postupech – vykonávání elementárních kroků, vstupu, výstupu, rozhodování a jejich propojování (v základních algoritmech lze vše ostatní složit z těchto prvků).
- Doménově závislé znalosti o elementárních krocích použitelných v dané doméně a obvykle vázaných na cílový systém implementující algoritmus nebo umožňujících interakci s okolním světem (ve výuce programování tyto kroky vycházejí z možností daného programovacího jazyka).

Existence dvou kategorií potřebných znalostí může vést i k nutnosti vytvoření realizačního týmu složeného z odborníků na obecnou algoritmizaci a odborníků na doménu, ze které pochází řešený problém.

## 2 Podpora výuky algoritmizace

Proces algoritmizace lze podrobně zkoumat jeho rozdělením na dílčí typické úlohy - formulaci problému, analýzu úlohy, vytvoření algoritmu řešení a testování algoritmu. Z výše uvedeného však plyne, že toto klasické členění je při výuce nutné rozšířit a doplnit kroky směřujícími k vytvoření příslušných mentálních modelů a získání znalostí.

- Získání obecných znalostí o algoritmizaci
- Získání doménově závislých znalostí
- Formulace problému
- Analýza úlohy
- Vytvoření algoritmu řešení
- Testování algoritmu

Výuka algoritmizace by měla pokrýt všechny uvedené fáze s tím, že doménová a problémová závislost by měly být minimalizovány a důraz kladen na obecné znalosti. Zásadní je zde především

volba vhodného jazyka pro prezentaci algoritmů, ten pak významně ovlivňuje efektivitu výuky. V další části příspěvku se proto zaměříme na hlavní kategorie podpůrných nástrojů a jejich zástupce, tyto kategorie budeme charakterizovat právě podle způsobu prezentace algoritmu:

- Přímá prezentace pomocí programovacího jazyka
- Použití symbolických (grafických) komponent s programovacím jazykem v pozadí
- Prezentace vizuálními jazyky jako jsou např. vývojové diagramy
- Prezentace pomocí pseudokódu
- Prezentace pomocí přirozeného jazyka

Je jasné, že celková efektivita výuky bude záviset i na předchozích znalostech a zkušenostech studentů, tedy cílové skupiny. Pro následující porovnání byla proto zvolena cílová skupina začátečníků bez předchozích znalostí tématu či témat příbuzných, tedy skupina, jejímž cílem není následné užití znalostí ve výuce programování, ale spíše pochopení principů řešení úloh pomocí algoritmů.

Aby bylo možné uvedené kategorie alespoň rámcově porovnat, bylo nutné stanovit kritéria pro toto porovnání. Ta byla zvolena a také vyhodnocována především s ohledem na zvolenou cílovou skupinu, která bude mít na podpůrné nástroje specifické požadavky:

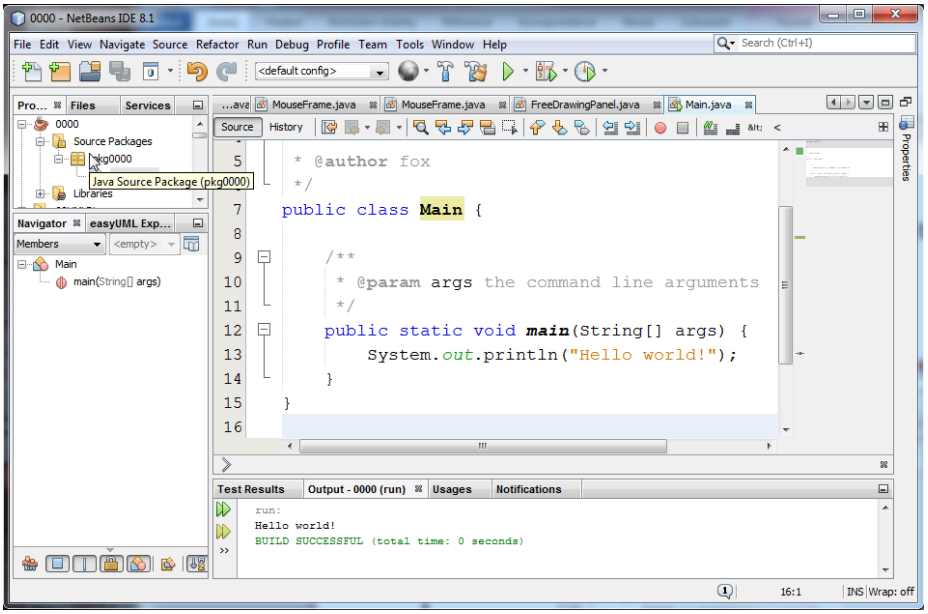
- Jednoduchá obsluha nástroje – cílová skupina má často pouze základní znalosti IT.
- Minimální nároky na další s algoritmizací nesouvisející nutné znalosti – tyto znalosti vytvářejí bariéru pro užití daného nástroje.
- Podpora práce s elementárními komponentami dané domény – cílová skupina může chtít použít algoritmizaci i na jiné domény, než nejčastěji uváděné (např. IT a programování).
- Standardizace popisu algoritmu – zde použito především ve smyslu užití nástrojů, které cílové skupině umožní opakované užití algoritmů a zajistí možnost komunikace o nich.
- Názornost prezentace algoritmu – názorná forma prezentace usnadní cílové skupině pochopení problematiky.
- Podpora testování algoritmu – v souladu s Bloomovou hierarchií jde o významnou podporu, díky ní může cílová skupina korigovat své znalosti a lépe si je uchovat.

## 3 Porovnání nástrojů pro podporu výuky algoritmizace

V následujícím textu budou představeny výše popsané kategorie a jejich zástupci. Uvedené hodnocení vychází ze zkušeností autora během více než 20 leté praxe ve výuce ICT, programování a algoritmizace.

*Přímá prezentace pomocí programovacího jazyka*

Tato skupina nástrojů bývá v praxi používána nejčastěji, zejména v situacích, kdy je výuka algoritmizace přímo propojena s výukou programování (což však není nutné a často ani vhodné). Za typické představitele je v této kategorii možno považovat všechna integrovaná prostředí pro vývoj v různých programovacích jazycích (IDE). Příkladem může být např. prostředí NetBeans od firmy Oracle, primárně zaměřené na vývoj v Javě (2).



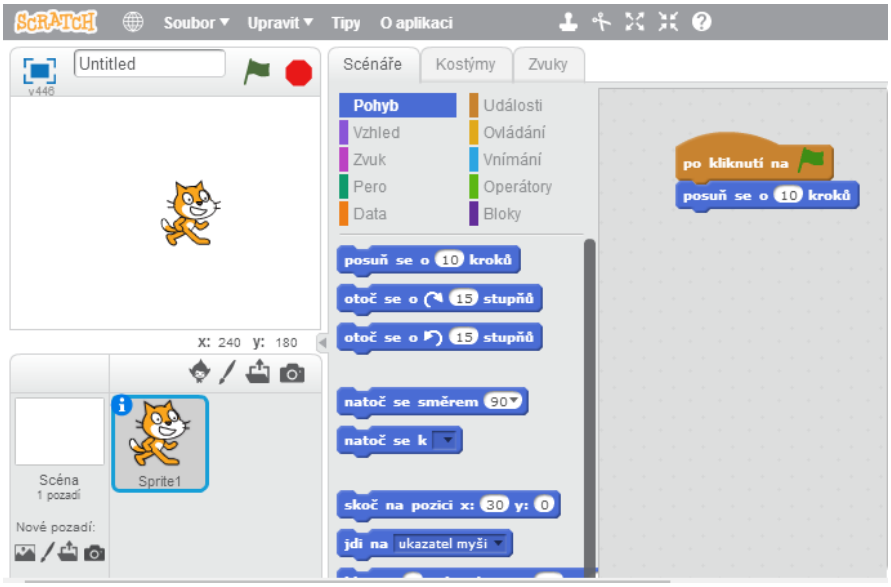

*Obr. 2 Integrované prostředí NetBeans (zdroj: vlastní)*

Jak je zřejmé z obrázku, toto IDE poskytuje řadu funkcí pro vývoj a testování programů, podpora algoritmizace je víceméně „vedlejším efektem“. Samotné přímé použití zvoleného programovacího jazyka klade na uživatele další nároky na znalost tohoto jazyka a pochopení jeho základních principů. Doménové elementární komponenty jsou zde přímo používány (konstrukty programovacího jazyka), využití pro jinou (ne přímo podporovanou) doménu je možné jen v rámci její implementace v daném jazyce. Standardizace popisu algoritmu je plně podporována syntaxí jazyka. Názornost vytvořeného algoritmu není nijak vysoká, pro plné pochopení je opět nutná dobrá orientace v programovém kódu. Velmi dobré jsou však nástroje pro testování algoritmu, a to jak na úrovni syntaktické správnosti, tak na úrovni ověření logického návrhu a při běhu programu.

Doplňkovým efektem užití IDE je doplnění znalostí o daném programovacím jazyce, což však může být zejména pro některé skupiny studentů či žáků zbytečné. Dalšími nástroji z této kategorie by bylo např. prostředí Elipse (3) či další specifičtěji zaměřená IDE.

*Použití symbolických (grafických) komponent*

V souladu s celkovou logikou užití jsou v této kategorii dostupná řešení pro méně znalé uživatele s větším důrazem na algoritmickou stránku úloh. Typickými představiteli jsou pak nástroje jako např. Snap! (4) či Scratch (5).




*Obr. 3 Nástroj Scratch (zdroj: vlastní)*

Již podle obrázku je zřejmé, že ovládání aplikace je velmi jednoduché a po krátkém úvodu jej zvládne i začátečník. Z hlediska dalších nároků na znalosti je potřeba říci, že např. Scratch je zaměřen na událostmi řízené programování (vytvořené algoritmy jsou zde také nazývány scénáře) a obsahuje i prvky objektového přístupu. Obě tyto oblasti jsou však natolik logicky zapracovány, že práce s nimi je pro uživatele vcelku přirozená. Zapracovány jsou i další funkce prostředí, které také s algoritmizací přímo nesouvisí (např. možnost grafického dopracování aktérů).

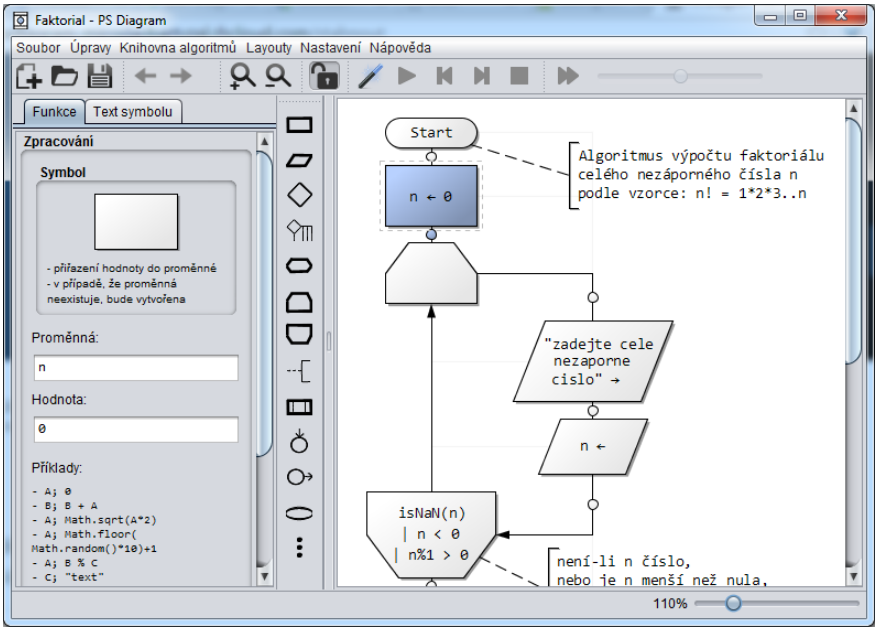
Prostředí se omezuje pouze na jednu doménu (popis scény a chování aktérů v ní), podobně jako klasické programovací jazyky však umožňuje i návrh vlastních komponent, které mohou základní sadu komponent rozšířit. Standardizace popisu je zde dána jako u předchozí kategorie jasnými pravidly skladby a množinou použitelných komponent. Vysoká názornost popisu algoritmu je podpořená i barevným odlišením komponent a konstruktů z různých oblastí. Podpora testování algoritmu je velmi dobrá ve fázi kompozice (syntaktická kontrola), výrazně slabší však při ověřování logiky algoritmu nebo jeho postupném krokování.

Zajímavostí je použití tohoto typu nástroje i pro vývoj jednodušších, avšak plnohodnotných aplikací pro OS Android s názvem MIT App Inventor (6).

*Prezentace vizuálními jazyky jako např. vývojovými diagramy*

V této kategorii nástrojů lze rozlišit i její podkategorie. První z nich jsou nástroje čistě pro vizuální popis algoritmu jako např. MS Visio (7). Tyto nástroje jsou však spíše grafickými editory, než podporou pro výuku algoritmizace. Jejich přínos spočívá v omezení použitelných komponent na ty používané např. ve vývojových diagramech. Podobnou charakteristiku mají i nástroje zaměřené na standard Unified Modeling Language (UML) a v něm užívané diagramy aktivit. Jejich výhodou může být adaptace i na jiné domény.

Druhou podkategorii tvoří již plnohodnotné nástroje s výrazně většími možnostmi. Pro následující charakteristiku této kategorie byl zvolen český produkt PS Diagram (8).




*Obr. 4 Nástroj PS Diagram (zdroj: vlastní)*

Pro nástroje této kategorie je typické přímé zacílení na užití daného grafického jazyka, zde vývojových diagramů. Prostředí je velmi jednoduché a bez zbytečných dalších funkcí, pro zvládnutí kterých by byly potřebné další znalosti. Pouze pokud budeme používat nástroj v plném rozsahu, je nutné pochopit základní datové struktury, stejně jako u předchozích kategorií. Plné možnosti programu (včetně podpory testování) využijete pouze na standardní doméně spojené s tvorbou programů, nástroj, ale umožňuje i návrh vlastních textově popsaných komponent. Pro popis algoritmu jsou použity vývojové diagramy, standardizace popisu je tedy zřejmá. Velmi vysoká je i názornost prezentace algoritmu, a to jak při návrhu, tak při běhu, což nástroj rovněž umožňuje. Na velmi vysoké úrovni je i podpora testování během tvorby (syntaxe), tak především při běhu a krokování navrženého algoritmu.

*Prezentace pomocí pseudokódu*

Následující dva způsoby prezentace algoritmů jsou také používány, nicméně jejich podpora specializovanými nástroji ne velmi malá. První a poslední sledované kritérium bylo pro následující oblasti nastaveno dle běžně používaného textového editoru, který cílová skupina dobře zná, ale nemá žádnou podporu pro testování algoritmů.

Při zkoumání prezentace pomocí pseudokódu se bohužel podařilo najít pouze jediný nástroj, který by ji využíval a podporoval algoritmizaci na vyšší úrovni, než syntaktické. Šlo o nadstavbu Code Rocket (9) nad MS Visual Studio. Jedná se však o specifickou syntaxi a užití. Příklady pseudokódu jsou uvedeny na obr. 5.

```
Input: start, goal(n), h(n), expand(n)
Output: path
if goal(start) = true then return makePath(start)

open ← start
closed ← ∅
while open ≠ ∅ do
    sort(open)
    n ← open.pop()
    kids ← expand(n)
    forall the kid ∈ kids do
        kid.f ← (n.g + 1) + h(kid)
        if goal(kid) = true then return makePath(kid)
        if kid ∩ closed = ∅ then open ← kid
    closed ← n
return ∅
```

```
Initialize n_Rounds and n_Steps
Compute threshold sequence τ_r
Randomly generate current solution x^c ∈ 𝒳
for r = 1 : n_Rounds do
  for i = 1 : n_Steps do
    Generate x^n ∈ 𝒩(x^c) and compute Δ = f(x^n) − f(x^c)
    if Δ < τ_r then x^c = x^n
  end for
end for
x^sol = x^c
```

*Obr. 5 Ukázky pseudokódu (zdroj: https://github.com/libgdx/gdx-ai/wiki/A* a http://comisef.wikidot.com/concept:thresholdaccepting)*

Při tvorbě pseudokódu je prakticky nutností znalost některého programovacího jazyka, bez ní je obtížně pochopitelný. Naproti tomu je dobře použitelný pro různé domény. Standardizace popisu však neexistuje, jak dokumentují uvedené ukázky. To také brání širšímu uplatnění této reprezentace, která je vhodná zejména pro ty, kteří již algoritmizaci ovládají a budou algoritmus implementovat do některého z programovacích jazyků. Podpora testování vzhledem k absenci konkrétních nástrojů neexistuje.

*Prezentace pomocí přirozeného jazyka*

Při prezentaci algoritmu přirozeným jazykem jsou zásadními problémy především nejednoznačnost a velká variabilita jazyka bez existujících omezení v dané oblasti, což společně s nejednotným názvoslovím zabraňuje standardizaci. Naopak je tato forma přístupná komukoliv bez dalších nutných znalostí (mimo použité názvosloví) a problematika se nemusí omezovat na konkrétní doménu. Názornost popisu algoritmu je nízká a závisí i na subjektivní interpretaci textu.

## 4 Vyhodnocení porovnání

Pro hodnocení jednotlivých nástrojů byla použita kritéria navržená výše a posuzovaná z hlediska výuky algoritmizace pro cílovou skupinu začátečníků. Váha jednotlivých kritérií byla odhadnuta na základě dlouholetých zkušeností s výukou v této oblasti a významu kritéria pro dosažení výukového cíle. Hodnocení jednotlivých produktů a kritérií probíhalo jako ve škole na stupnici 1 - 5. Výsledné hodnocení je dáno váženým průměrem.

*Tabulka 1 Porovnání nástrojů pro podporu výuky algoritmizace (zdroj: vlastní)*

| Kritérium | Váha | NetBeans | Scratch | PS Diagram | Pseudo-kód | Přirozený jazyk |
|---|---|---|---|---|---|---|
| Jednoduchá obsluha | 1 | 4,0 | 2,0 | 1,0 | 1,0 | 1,0 |
| Nároky na další znalosti | 2 | 5,0 | 3,0 | 1,5 | 2,0 | 1,0 |
| Doménové komponenty | 2 | 4,0 | 3,5 | 2,5 | 1,0 | 1,0 |
| Standardizace popisu | 3 | 1,0 | 1,0 | 1,0 | 5,0 | 5,0 |
| Názornost prezentace | 3 | 4,0 | 1,0 | 1,0 | 3,0 | 4,5 |
| Podpora testování | 3 | 1,0 | 3,0 | 1,0 | 5,0 | 5,0 |
| **Hodnocení** | | **2,86** | **2,14** | **1,29** | **3,29** | **3,46** |

Z uvedených výsledků vyplynuly jasné závěry týkající se volby vhodného podpůrného nástroje pro výuku algoritmizace u cílové skupiny začátečníků. Je jím produkt PS Diagram, který je na uvedenou oblast přímo zaměřen a jehož užití s sebou nese nejmenší vstupní bariéry. To odpovídá i praktickým zkušenostem z výuky.

## Závěr

Cílem tohoto příspěvku bylo zamyslet se nad volbou možných nástrojů pro podporu výuky základů algoritmizace u cílové skupiny začátečníků bez předchozích znalostí. Po úvodu do problematiky bylo provedeno rozdělení nástrojů do kategorií, založené na způsobu reprezentace algoritmů. Tyto kategorie byly následně porovnány podle navržených kritérií s ohledem na zvolenou cílovou skupinu a dosažení výukového cíle, kterým je naučit cílovou skupinu obecným základům algoritmizace.

Z porovnání výsledků vyplynulo, že ideální formou podpory je využití nástroje, který je na danou oblast přímo zacílen a má nejmenší bariéry užití. Tím byla skupina nástrojů využívající pro prezentaci algoritmu vizuální jazyky, konkrétním hodnoceným nástrojem pak byl český produkt PS Diagram s funkcionalitou přesně odpovídající zaměření.

## Literatura

1. ANDERSON, Lorin W., KRATHWOHL, David R. a BLOOM, Benjamin S. *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives.* Allyn & Bacon, 2001.
2. ORACLE. NetBeans IDE 8.0.2 Release Information. [Online] 2016. [Citace: 05. 05 2016.] Dostupné z: https://netbeans.org/community/releases/80/.
3. THE ECLIPSE FOUNDATION. Eclipse - The Eclipse Foundation open source community website. [Online] 2016. [Citace: 05. 05 2016.] Dostupné z: https://eclipse.org/.
4. MIT MEDIA LAB. Snap! (Build Your Own Blocks) 4.0. [Online] 2016. [Citace: 05. 05 2016.] Dostupné z: http://snap.berkeley.edu/.
5. MIT MEDIA LAB. Scratch - Imagine, Program, Share. [Online] 2016. [Citace: 05. 05 2016.] Dostupné z: https://scratch.mit.edu/.
6. MIT. MIT App Inventor. [Online] 2016. [Citace: 05. 05 2016.] Dostupné z: http://appinventor.mit.edu/explore/.
7. MICROSOFT. Profesionální software pro grafy a diagramy | Microsoft Visio. [Online] 2016. [Citace: 05. 05 2016.] Dostupné z: https://products.office.com/cs-cz/visio/flowchart-software.
8. BARTYZAL, Miroslav. PS Diagram. [Online] 2016. [Citace: 05. 05 2016.] Dostupné z: http://www.psdiagram.cz/.
9. RAPID QUALITY SYSTEMS. Code Rocket. [Online] 2016. [Citace: 05. 05 2016.] Dostupné z: http://www.rapidqualitysystems.com/.

## Kontaktní adresa:

Jiří Jelínek, Ing. CSc.,
Katedra informatiky a přírodních věd, Ústav technicko-technologický,
Vysoká škola technická a ekonomická v Českých Budějovicích,
Okružní 517/10, 370 01 České Budějovice, ČR, tel.: +420 387 842 186,
email: jelinek.vs@gmail.com