Ďalšie vzdelávanie učiteľov základných škôl a stredných škôl v predmete informatika

ŠTÁTNY PEDAGOGICKÝ ÚSTAV
NATIONAL INSTITUTE FOR EDUCATION

Ďalšie vzdelávanie učiteľov základných škôl a stredných škôl v predmete informatika

Didaktika programovania 1

Predmet: Didaktika programovania

Línia: Didaktika informatiky a informatickej výchovy

Operačný program VZDELÁVANIE

EURÓPSKA ÚNIA
Európsky sociálny fond

Európska únia
Európsky sociálny fond

Moderné vzdelávanie pre vedomostnú spoločnosť/Projekt je spolufinancovaný zo zdrojov ES

Didaktika programovania 1

Identifikácia modulu

Aktivita projektu:	1.3 Ďalšie vzdelávanie kvalifikovaných učiteľov informatiky na 2. stupni ZŠ a na SŠ
Línia aktivity:	Didaktika informatiky a informatickej výchovy
Predmet:	Didaktika programovania

Garant predmetu:

RNDr. Ľubomír Salanci, PhD.
KZVI FMFI UK, Bratislava
salanci@fmph.uniba.sk

Autori:

RNDr. Ľubomír Salanci, PhD., KZVI FMFI UK, Bratislava
PaedDr. Monika Tomcsányiová, PhD., KZVI FMFI UK, Bratislava
RNDr. Andrej Blaho, KAI FMFI UK, Bratislava

Zaradenie modulu

Modul Didaktika programovania 1 je prvým modulom, ktorý sa venuje téme, akým spôsobom učiť programovanie na druhom stupni základnej a na strednej škole.

Abstrakt modulu

Vyučovanie programovania je najdôležitejšou súčasťou informatiky. Je nevyhnutné, aby sa účastníci vzdelávania dozvedeli o postupnosti tém a o metodike toho, ako učiť programovať svojich žiakov na základnej alebo strednej škole. Tiež je dôležité, aby si aj prakticky vyskúšali navrhovať vyučovacie hodiny k jednotlivým témam.

Tento modul sa viac venuje vyučovaniu programovania na základnej škole. Dozvieme sa tiež o histórii informatiky a programovania, spomenieme rôzne detské programovacie prostredia a budeme pripravovať vyučovacie hodiny pre žiakov na 2. stupni ZŠ. Na tento modul potom nadväzuje modul Didaktika programovania 2, ktorý bude venovaný téme programovania na strednej škole.

Obsah

Úvod

V tomto module sa venujeme didaktike programovania na základnej a strednej škole. Didaktikou programovania rozumieme vednú oblasť, v ktorej skúmame a hľadáme spôsoby, ako učiť algoritmy a programovanie. Snažíme sa porozumieť poznávaciemu procesu, tomu, ako vzniká nový poznatok, ale aj ťažkostiam, ktoré majú žiaci s porozumením algoritmov alebo s programovaním. Pritom chceme objavovať zákonitosti alebo pravidlá, ktoré by prispeli k lepšiemu vyučovaniu informatiky.

Pre tento modul bude potrebné, aby mali účastníci k dispozícii nasledujúce softvérové prostredia: Cirkus šaša Tomáša, Karel, Baltík, Panák, Živý obraz, Izy Logo, Imagine, Scratch. Tiež je nevyhnutné, aby bola pre každú 5-6 člennú skupinu účastníkov vzdelávania dostupná učebnica Tvorivá informatika 1. zošit z programovania. Výučba bude prebiehať v miestnosti s dataprojektorom, dostatočne veľkou tabuľou.

Cieľ modulu

Cieľom modulu je, aby účastník vzdelávania:

- Porozumel postaveniu, úlohe a cieľom programovania v informatike na ZŠ a SŠ.
- Aby dokázal analyzovať tematické celky z programovania na ZŠ.
- Vedel navrhnúť a zostaviť vyučovacie hodiny programovania pre ZŠ.

Vstupné vedomosti

Požadované prerekvizity

Účastník vzdelávania musí mať absolvované všetky moduly týkajúce sa programovania, t.j. Programovanie 1 až 3, Programovanie 4 (Pascal) alebo Programovanie 4 (Imagine).

Predpokladané vstupné vedomosti, skúsenosti a zručnosti

Pozná aspoň tieto programovacie jazyky: Pascal (Delphi / Lazarus), Logo (Imagine).

Kapitola 1: Maturita (riešenie monitora)

Maturitu považujeme za cieľovú métu stredoškolského štúdia časti študentov. Tento modul – Didaktiku programovania – však maturitou práve začíname. Viedli nás k tomu nasledujúce dôvody:

- Maturita z informatiky vymedzuje (by mala vymedzovať) obsah, ktorý považujeme za dôležitý z pohľadu informatiky ako vedného odboru.
- Je potrebné poznať, kam vyučovanie informatiky a programovania smeruje od ZŠ po SŠ.
- Dá sa vidieť, ktoré kompetencie (poznatky, schopnosti) z informatiky považujeme za dôležité (v súčasnosti a blízkej budúcnosti).

Preto sa domnievame, že s obsahom maturít by mal byť oboznámený nielen učiteľ informatiky strednej školy, ale aj učiteľ, ktorý učí na základnej škole. Tak aspoň pozná cieľ (aj keď dlhodobý a vzdialený), pre ktorý pripravuje a vzdeláva svojich žiakov.

Spoznanie maturít a riešenie úloh z programovania sa nám zdalo ako vhodný prechod medzi programovaním, ktoré sme doposiaľ zažívali a didaktikou programovania. Riešením maturitných úloh sa postupne nastavíme na úroveň, ktorá je vyžadovaná na strednej škole, na konci štvrtého – maturitného ročníka.

Úlohy, ktoré nájdeme v maturitách z informatiky, môžeme rozdeliť na dve časti:

- Prvú časť tvoria tzv. **neprogramátorské** úlohy.
- Druhú časť tvoria tzv. **programátorské** úlohy.

Počet neprogramátorských a programátorských úloh je zhruba v pomere 1:1.

Neprogramátorská časť maturity

Neprogramátorská časť maturít obsahuje úlohy, ktoré nevyžadujú znalosti programovania.

Aktivita 1.1	Prečítajte si jednotlivé úlohy. Vedeli by ste ich vyriešiť?

Ukážky úloh z prvej časti maturitného monitora z roku 2004 (zdroj Štátny pedagogický ústav):

03 Pojmom *Open Source* sa označuje softvér, ktorý

(A) má otvorený kód, t. j. ide o nedokončený program.
(B) má zverejnené aj zdrojové kódy a ktorý môžeme ľubovoľne upravovať a ďalej šíriť.
(C) je možné používať len v otvorených počítačových systémoch.
(D) iným menom nazývame tiež freeware.

05 Antivírový program ohlásil, že v pamäti počítača je vírus. Ak nechceme prísť o dáta uložené na disku, je najvhodnejšie

(A) zazálohovať všetky súbory, naformátovať disk a znovu nainštalovať operačný systém.
(B) okamžite spustiť liečenie antivírovým programom nainštalovanom v počítači.
(C) reštartovať počítač z nezavírenej diskety a potom spustiť liečenie antivírovým programom.
(D) reštartovať počítač a spustiť liečenie antivírovým programom nainštalovaným na disku počítača.

10 Naskenovali sme stránku textu napísaného v bežnom textovom editore, ktorú sme chceli publikovať na webe. Všimli sme si, že autor mal problémy s pravopisom, preto sme sa rozhodli, že v naskenovanom súbore pred jeho uverejnením na webe opravíme pravopisné chyby. Ktoré z nasledujúcich postupov sú na to vhodné?

(1) Naskenovaný súbor otvoríme v bitmapovom grafickom editore a v ňom vygumujeme nesprávne písmená a nakreslíme správne.

(2) Naskenovaný súbor otvoríme v textovom editore a použijeme funkciu *nájdi a nahraď*.

(3) Na naskenovaný súbor použijeme program na rozpoznávanie textu a po ukončení rozpoznávania v texte opravíme chyby.

(4) Naskenovaný súbor otvoríme vo vektorovom grafickom editore a použijeme funkciu *nahraď tvar iným tvarom*.

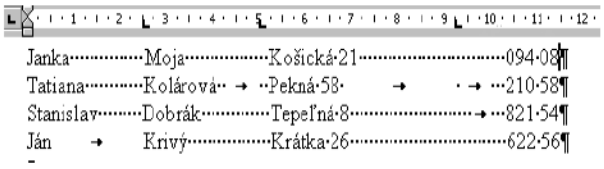
(A) Iba postupy (1) a (2). **(B)** Iba postupy (1) a (3).

(C) Iba postupy (3) a (4). **(D)** Iba postupy (2) a (4).

16 Koľko MB pamäte zaberie súbor s minútovou nahrávkou skladby s týmito charakteristikami: 44 100 Hz, 16-bitové stereo, kompresný pomer 0,2? (Výsledok zaokrúhlite na 1 desatinné miesto.)

Text k úlohám 24 a 25

Autor tohto dokumentu nepozná základné zásady písania elektronického textu. Namiesto použitia tabulátora sa snažil písať do stĺpcov a pomáhal si pritom vkladaním medzier. Pretože súbor má veľa riadkov, trvalo by dlho opraviť manuálne všetky riadky.



(znak „ ¶ “ zastupuje enter, znak „ → “ zastupuje tabulátor, znak „ . “ zastupuje medzeru)

24 Tu sú štyri možné postupy úpravy pôvodného textu:

(1) Zmena pravého okraja na 12 cm.
(2) Zmena veľkosti písma.
(3) Zmena fontu z proporcionálneho na neproporcionálny.
(4) Zmena riadkovania.

Pri ktorých dvoch z nich by sa porušilo zarovnanie do stĺpcov?

(A) Pri postupoch (1) a (2). **(B)** Pri postupoch (2) a (3).

(C) Pri postupoch (3) a (4). **(D)** Pri postupoch (1) a (4).

Môžeme si všimnúť, že neprogramátorské úlohy sú zamerané na testovanie kompetencií z oblasti:

- spracovania informácii pomocou softvérových nástrojov,
- reprezentácie údajov,
- počítačových systémov, bezpečnosti alebo spoločenských aspektov informatiky.

Aktivita 1.2	Diskutujte o jednotlivých zadaniach. Majú jednoznačné riešenie?
Aktivita 1.3	Pomenujte, ktorú oblasť informatiky testujú úlohy na predchádzajúcich ukážkach.

Programátorská časť maturity

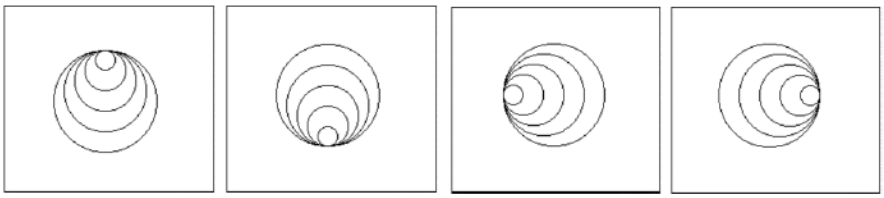
Programátorská časť maturity obsahuje úlohy, ktoré sú zamerané na algoritmické myslenie a programovanie. Tieto úlohy preskúmame podrobnejšie. Najskôr ich ale všetky vyriešime.

Aktivita 1.4	Vyriešte programátorské úlohy z maturitných testov. Vytvorte 5 až 6 členné pracovné skupiny. Každá skupina dostane zadania úloh z programovania, ktoré sú súčasťou maturít (maturitného monitora). V skupine môžete spolupracovať. Zadeľte si prácu tak, aby každý člen v skupine niečo riešil a aby ste spoločne stihli vyriešiť všetky úlohy z programovania.

Ukážky z druhej časti maturitného monitora z roku 2004, (zdroj Štátny pedagogický ústav):

30 Príkaz `Circle (x, y, r)` nakreslí kružnicu, ktorá má polomer *r* a ktorej stred má súradnice *x*, *y*. Ktorý z uvedených obrázkov by bol výstupom nasledujúceho cyklu?

```
for i:= 1 to 5 do Circle(100 + i * 10, 100, i * 10);
```



(A) **(B)** **(C)** **(D)**

32 Ktorý z nasledujúcich algoritmov napíše najviac hviezdičiek?

```
i:= 1;
j:= 1;
while (i <= 5) and (j <= 3) do
  begin
    write('*');
    i:= i + 1;
    j:= j + 1;
  end;
```

(A)

```
i:= 1;
j:= 1;
while (i <= 5) or (j <= 3) do
  begin
    write('*');
    i:= i + 1;
    j:= j + 1;
  end;
```

(B)

```
i:= 1;
j:= 1;
while not (i <= 5) or (j <= 3) do
  begin
    write('*');
    i:= i + 1;
    j:= j + 1;
  end;
```

(C)

```
i:= 1;
j:= 1;
while (i <= 5) and not(j <= 3) do
  begin
    write('*');
    i:= i + 1;
    j:= j + 1;
  end;
```

(D)

Text k úlohám 39 – 41

Pri tréningu sa skokan do výšky riadi nasledujúcim algoritmom:

```
nastav pociatocnu vysku 180 cm.
opakuj
  Ak skocis vysku na prvy pokus, tak zvys latku o 5 cm.
  Ak skocis vysku na druhy pokus, tak zvys latku o 3 cm.
  Ak skocis vysku na treti pokus, tak zvys latku o 1 cm.
pokial budes mat 3 neuspesne pokusy na jednej vyske.
```

Postupnosť pokusov budeme v ďalšom zapisovať pomocou písmen p – preskočil a n – nepreskočil.

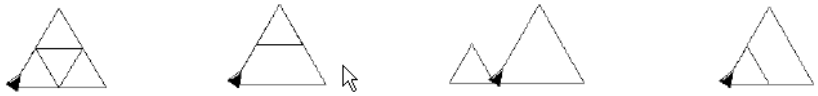
39 Akú najvyššiu výšku preskočil skokan, ak postupnosť jeho pokusov bola: p p n n p p n p n n n?

40 Skokan postupne skákal výšky 180 cm, 181 cm, 182 cm, 187 cm, 192 cm, 195 cm, 196 cm. Zapíšte pomocou písmen p, n jeho postupnosť pokusov.

41 Aký najmenší počet pokusov mohol mať pretekár, ak posledná výška, ktorú skákal, bola 192 cm a na ďalšiu sa už nedostal?

42 Príkaz `troj(s: real);` nakreslí rovnostranný trojuholník so stranou dĺžky *s*, pričom poloha pera a jeho natočenie na konci sú tie isté ako pred vykonaním tohto príkazu. Aký obrázok nakreslí nasledujúca postupnosť príkazov?

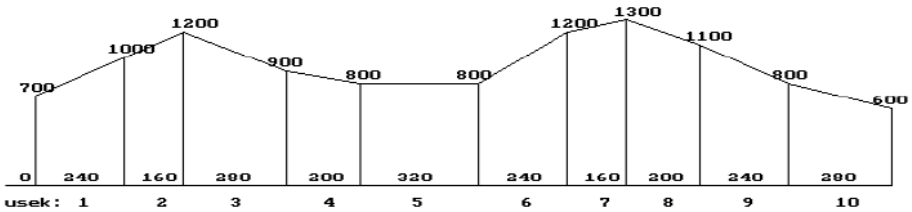
```
troj(100); dopredu(50); vpravo(60); troj(50); vlavo(60); dopredu(-50);
```



(A) (B) (C) (D)

Text k úlohám 45 – 50

V poliach `x, y: array[0..100] of integer;` *sú zaznamenané údaje o N úsekoch turistickej trasy (1 < N <= 100). Zaznamenané sú body na trase, v ktorých sa mení sklon terénu (prudšie stúpa, začína klesať, ...). V x[0], y[0] sú súradnice začiatku, a platí x[0] = 0. V x[i] je posun v smere osi x oproti predchádzajúcemu stanovisku, v y[i] je nadmorská výška i-teho bodu na trase. Úsek i má na osi x krajné body x[i – 1] a x[i]. Údaje sú v metroch.*



45 Napíšte príkaz, ktorý je potrebné doplniť na miesto {1} tak, aby program na konci vypísal dĺžku celej trasy na osi *x* v metroch.

```
{var dlzka, i: integer;}

dlzka:= 0;
for i:= 1 to N do  {1}  ;
writeln('dlzka trasy v smere osi x: ', dlzka, ' m');
```

Úlohy testujú rôzne kompetencie (schopnosť porozumieť cudziemu algoritmu, schopnosť trasovať program, ...), robia to rôznym spôsobom (treba zistiť, čo program robí, treba doplniť program tak, aby čosi vykonal,...), a tiež sú zamerané na rôzne oblasti alebo témy (cyklus, polia, náhodné čísla a podobne).

Aktivita 1.5	Pomenujte tému alebo oblasť z programovania, ktorej sa jednotlivé úlohy týkajú.
Aktivita 1.6	Povedzte, aké kompetencie a akým spôsobom jednotlivé úlohy testujú.

Kapitola 2: História vyučovania informatiky

Vyučovanie informatiky má na Slovensku už niekoľko desaťročnú tradíciu. Môžeme povedať, že siaha až do šesťdesiatych rokov minulého storočia..

Aktivita 2.1	Povedzte, kedy sa na vašej škole začala učiť informatika
Aktivita 2.2	Čím bola charakteristická informatika v 60., 70., 80., 90. rokoch? A čím je charakteristická v súčasnosti? Aký bol obsah informatiky v jednotlivých obdobiach? Čo sa v rámci informatiky učilo?

Obsah predmetu informatika sa vyvíjal a v priebehu minulých desaťročí výrazne menil. Dá sa pozorovať, že obsah ako aj vyučovanie informatiky bolo podriadené vývoju počítačov a technológií.

60. a 70. roky

V 60. a 70. rokoch boli počítače pre verejnosť v podstate nedostupné. Informatika ako všeobecnovzdelávací predmet na základnej ani strednej škole neexistovala. Existovalo však niekoľko málo gymnázií, ktoré mali triedy so zameraním na programovanie (napr., na Gymnáziu Jura Hronca). Programy sa zväčša písali na papier. Ak bola možnosť, program niekto prepísal na dierne štítky a spustil na minipočítači alebo sálovom počítači. Výsledky aj chyby v programe sa žiaci dozvedeli niekedy až o týždeň. Na Slovensku sa didaktika informatiky v tomto období rodila.

Aj v zahraničí uvažovali nad tým, ako sprístupniť programovanie začiatočníkom – spočiatku dospelým vysokoškolským študentom. Napríklad, už v roku 1963 bol navrhnutý programovací jazyk BASIC a jazyk Pascal vznikol v rokoch 1968–1969 s cieľom získavať dobré programátorské návyky. V roku 1967 však vznikol jazyk Logo, ktorý autori pôvodne navrhli pre účely hravého vyučovania matematiky. Postupne sa Logo sa stalo jazykom – prostredím, ktoré sprístupnilo počítače a programovanie deťom.

80. roky

V 80. rokoch sa rozrástol trh s počítačmi. Vznikajú domáce počítače a mikropočítače. V porovnaní so súčasnými počítačmi boli stotisíckrát pomalšie a mali približne toľkokrát menej pamäte. Aplikácie, ktoré dnes bežne poznáme a používame, ešte neexistovali, prípadne sa v tomto období začínali postupne vytvárať.

Napriek akej takej dostupnosti boli počítače v tomto období raritou. V rodinách, ktoré počítače vlastnili, sa prevažne používali na hranie hier. Aj vzhľadom na nedostatok aplikácii nebolo väčšine populácie jasné, na čo môžu byť počítače užitočné. Prevažne sa vnímali ako nástroj na hranie sa hier, zriedka ako vec, ktorá sa dá programovať.

V priebehu 80. rokov sa informatika stáva povinným predmetom na gymnáziách. V tomto období prevládali na školách zväčša 8-bitové počítače PMD 85, Didaktik Alfa, niekde aj PP01. Vyučovanie informatiky bolo poznačené nedostatkom aplikácii. Keďže tie neboli k dispozícii, predpokladalo sa, že si ľudia budú aplikácie sami programovať. Informatika sa v tomto období prevažne stotožnila s vyučovaním programovania v jazyku BASIC. Zjednodušene by sme mohli povedať, že informatika = algoritmy a programovanie.

Zaujímavé je sledovať aj typy úloh alebo problémov, ktoré sa v tomto období riešili

v rámci Informatiky. Prevažovali textovo orientované úlohy (úlohy, ktoré podľa zadaného vstupu vypisujú na obrazovku texty, čísla), numerické výpočty, programovanie jednoduchej textovej databázy. S korytnačkou Žofkou (zjednodušený jazyk Logo) alebo prostredím Karel sa deti prevažne stretli na programátorských krúžkoch.

90. roky

Koncom 80. a na prelome 90. rokov dobiehame „svet za oponou“. Navyše aj vo svete sú celé 90. roky zasiahnuté búrlivým vývojom a veľkými zmenami v informatike.

Začínajú sa používať 16 bitové počítače, ktoré sú onedlho nahradené 32 bitovými počítačmi. Menia sa operačné systémy: 8-bitové systémy sú nahradené systémom DOS, objavuje sa systém Windows 3.11, ktorý je nahradený systémom Windows 95 alebo ďalšími systémami. Začína sa éra internetu (e-mail, web). Začínajú sa hromadne používať nové zariadenia (myš, tlačiareň, skener, disketa, tablet). Vznikajú nové aplikácie T602, MS Word alebo web prehliadače.

V tomto období sa nemení iba hardvér a softvér, ale aj spôsob, akým pracujeme s počítačom. Počítač už netreba programovať, pretože je k dispozícii množstvo aplikácií. Internet a rozmanité aplikácie dávajú počítačom nový zmysel využitia. Je však potrebné naučiť sa počítač a jeho nové aplikácie ovládať.

Ako na tieto zmeny reaguje škola? Začína sa s vyučovaním jazyka Pascal. Neskôr, v druhej polovici 90. rokov sa objavuje Comenius Logo, ako prostredie pre mladších žiakov. Viac-menej sa však uvoľnenie atmosféry v spoločnosti prejavuje v snahe zmeniť obsah predmetu Informatika. Niektoré snahy viedli až k tomu, že vzniká odpor voči programovaniu (programovanie akoby upadlo do nemilosti) a vo vyučovaní informatiky prevláda používateľský prístup – teda učenie aplikácií. Teraz by sme zjednodušene mohli povedať, že informatika = aplikácie.

Je jasné, že mnohé zmeny sú potrebné. Na druhej strane sa však do vyučovania Informatiky zanášajú rôzne deformácie (napríklad, učenie sa klávesových skratiek naspamäť, kopírovanie súborov naslepo v programe Norton Commander, výučba príkazov z ponuky textového editora T-602 a pod.)

Koniec predošlého a začiatok nového milénia, súčasnosť

Búrlivý rozvoj vo svete informačných technológií bol zohľadnený aj v nových osnovách predmetu Informatika. Osnovy vznikli v roku 1997. Výučba aplikácií (presnejšie, nástrojov na spracovanie údajov) aj výučba programovania sú súčasťou týchto osnov. V osnovách je ponechaná dostatočná voľnosť pre školu a učiteľa na to, aby rozsah vyučovania jednotlivých oblastí informatiky prispôsobili možnostiam školy aj svojich žiakov.

Okrem osnov bol postupne vypracovaný aj maturitný štandard z Informatiky. Z neho je zrejmé, že programovanie je pre informatiku veľmi dôležité a tvorí podstatnú časť maturity z Informatiky (až 50%).

S príchodom reformy školstva v roku 2008 sa Informatická výchova a Informatika stali povinnými predmetmi primárneho a nižšieho sekundárneho vzdelávania.

Aktivita 2.3	Aký priestor má vo vašom vyučovaní tematický okruh Postupy, riešenie problémov a algoritmické myslenie?

Zhrnutie

Obsah predmetu Informatika sa vyvíjal v závislosti od stavu technológií. Veľký posun nastal v tom, aké typy úloh a problémov sa v jednotlivých obdobiach riešili. Najskôr sa informatika stotožnila s programovaním, neskôr sa na programovanie na školách zanevrelo a informatika sa stotožnila s vyučovaním aplikácií. V súčasnosti je súčasťou informatiky aj výučba aplikácií aj programovania.

Mohli sme si všimnúť, ako predmet Informatika reagoval na vývoj technológií, ako prispôsobil svoj obsah zmenám alebo požiadavkám spoločnosti. Darilo sa to aj vďaka učiteľom, ktorí mali chuť skúmať, spoznávať nové technológie a paradigmy. Domnievame sa, že v tomto ohľade je informatika jedinečná. Takéto ohromné zmeny snáď žiaden iný predmet v poslednom období nezažil.

Kapitola 3: Detské programovacie prostredia

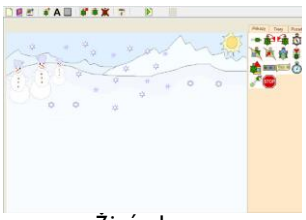
Ak plánujeme učiť programovanie žiakov na základne škole, zohráva výber vhodného programovacieho prostredia nesmierne dôležitú úlohu. Vo všeobecnosti platí, že čím mladších žiakov plánujeme učiť alebo čím väčšiu časť populácie chceme učiť, tým viac musíme byť pri voľbe prostredia aj programovacieho jazyka opatrní.

Ukážky detských prostredí

Detské programovacie jazyky a prostredia musia rešpektovať nielen určité **ergonomické kritériá**, ale bezpodmienečne musia rešpektovať aj **schopnosti detí**. Vieme napríklad, že deti do 12 rokov sú schopné rozmýšľať a vykonávať operácie s konkrétnymi predmetmi (tzv. obdobie konkrétnych operácií). Abstraktné myslenie (tzv. obdobie formálnych operácií) nastupuje a rozvíja sa až v neskoršom veku.

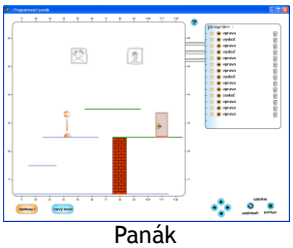
Cirkus šaša Tomáša

Detské prostredia preto vyzerajú veľmi rozdielne od prostredí, ktoré sú určené pre dospelých a profesionálnych programátorov.

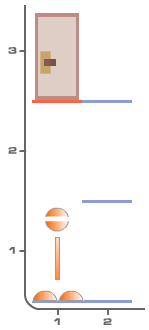


Živý obraz

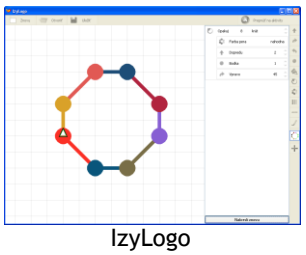
Definovanie správania sa objektov v Živom obraze.



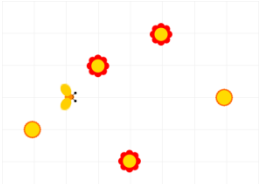
Panák



Postavičku treba naprogramovať tak, aby prišla k dverám



IzyLogo



V IzyLogu treba so včieklou postupne navštíviť všetky kvetiny

Aktivita 3.1

Pozrite si (prípadne vyskúšajte) niektoré prostredia:

- Cirkus šaša Tomáša
- Karel, Baltík, Panák
- Živý obraz
- Izy Logo
- Imagine

Pritom sa zamyslite a povedzte:

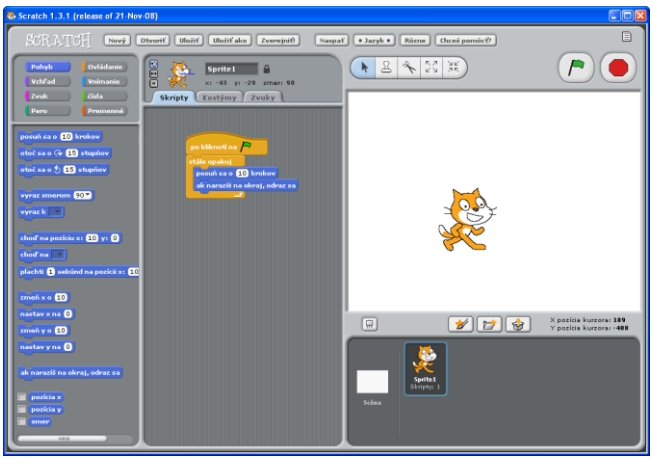
- Pre aký vek sú jednotlivé prostredia vhodné?
- Prečo prostredia vyzerajú tak, ako vyzerajú?
- Akým jazykom komunikujú s používateľom?
- Čo je cieľom prostredia – aké informatické kompetencie rozvíjajú (objavovanie algoritmu, porozumenie pravidlám, ovládanie objektov v priamom režime, zostavovanie poradia príkazov, trasovanie programu, riešenie elementárnych grafových alebo kombinatorických úloh, ...)?

Detské programovacie prostredia často využívajú rôzne **metafory**. Napríklad, korytnačka zastupuje (zosobňuje) grafické pero, ktoré kreslí relatívnym pohybom (t.j. nevyužíva súradnice). Vďaka metaforám ľahšie sprístupníme rôzne informatické koncepty, algoritmické konštrukcie, techniky aj mladším deťom.

Aktivita 3.2	Aké metafory používajú jednotlivé detské prostredia? Aké koncepty tieto metafory zastupujú, predstavujú?

Scratch

Scratch je detské programovacie prostredie, v ktorom vytvárame programy tým, že skladáme kartičky s príkazmi.



Ukážka z prostredia Scratch

Aktivita 3.3	Spustite Scratch a preskúmajte jeho prostredie.
Aktivita 3.4	Vytvorte program, ktorý po stlačení zelenej vlajočky spustí cyklické pochodovanie kocúra.
Aktivita 3.5	Vymenujte programové konštrukcie, ktoré ste objavili.
Aktivita 3.6	Porovnajte prostredia Imagine a Scratch: • Ako sa riadi pohyb hrdinov – korytnačky a mačky? • Aké výhody a aké nevýhody má zápis programov pomocou kartičiek? Porovnajte ho so zápisom programov v prostredí Imagine. • Akým spôsobom pracujete s premennými? • Na aké udalosti dokáže hlavný hrdina reagovať?

Zhrnutie

Aktivita 3.7	Porovnajte výhody a nevýhody detských programovacích prostredí s profesionálnymi jazykmi.

Kapitola 4: Programovanie v osnovách

Predmet informatika bol **donedávna** povinným iba na gymnáziách v prvom ročníku. Mal časovú dotáciou 2 hodiny týždenne:

1. stupeň ZŠ				2. stupeň ZŠ a 8 roč. gym.					Gymnáziá			
1.	2.	3.	4.	5.	6.	7.	8.	9.	I.	II.	III.	IV.
									2h			
									Informatika			

V prípade záujmu si niektoré školy mohli zostaviť študijný program s väčším počtom hodín informatiky. Výnimočne sa Informatická výchova a Informatika učili aj na niektorých základných školách.

V roku 2008 začala školská **reforma**. Nastáva preorganizovanie predmetov, školy dostávajú väčší priestor pri výbere predmetov, vzniká snaha o redukciu povinného učiva (toto sa ukazuje ako veľký problém pre ostatné predmety, keďže si každý myslí o svojom predmete, že je najdôležitejší).

Informatika sa po reforme stáva súčasťou vzdelávacej oblasti „matematika a práca s informáciami“. Stáva sa povinným predmetom aj na základnej škole:

1. stupeň ZŠ				2. stupeň ZŠ a 8 roč. gym.					Gymnáziá			
1	2	3	4	5	6	7	8	9	I	II	III	IV
	1h	**1h**	**1h**	**0,5h**	**0,5h**	**0,5h**	**0,5h**	**0,5h**	**1h**	**1h**	**1h**	
Informatická výchova				Informatika					Informatika			

V tabuľke vidíme, že niektorých ročníkoch sú hodinové dotácie na úrovni 0,5h za týždeň. Hodinové dotácie si môžu školy rozšíriť, napríklad 0,5h na 1h a podobne.

Aktivita 4.1

Ako ste si poradili s „nešikovnými“ hodinovými dotáciami v 5. až 9. ročníku na ZŠ?

Algoritmy a programovanie

Musíme dopredu upozorniť, že pojem programovanie používame v našom dokumente v posunutom zmysle slova. Totiž, definície pojmov „algoritmus“ a „programovanie“ hovoria zhruba nasledovné:

- Algoritmus = postup, návod na riešenie problému.
- Algoritmizácia = vytváranie (vymýšľanie a navrhovanie) algoritmov.
- Programovanie = prepis algoritmu do programovacieho jazyka.

Takéto striktné oddeľovania návrhu algoritmu od jeho prepisu do programovacieho jazyka malo v minulosti zmysel, pretože výpočtový čas bol drahý a neexistovali ani vhodné editory. V súčasnosti však takéto striktné oddeľovanie algoritmov od programovania stráca zmysel. Súčasné vývojové prostredia nám nielenže umožňujú algoritmy zapisovať priamo v programovacom jazyku, ale pomáhajú nám ich ladiť, vylepšovať (napr. profiler) alebo upravovať (napr. refaktorizácia).

Pre konštruktivistický prístup vo vyučovaní algoritmov a programovania však považujeme za nesmierne dôležité to, že naše nápady môžeme okamžite vyskúšať. Napríklad, v interpretovanom jazyku Imagine Logo píšeme príkazy do príkazového riadku. Výsledok vykonávania príkazov vidíme ihneď po stlačení klávesu Enter na obrazovke. Ešte ďalej ide v tomto smere prostredie IzyLogo. V ňom skladáme program z kartičiek, pričom zmena programu alebo aj parametra príkazu sa okamžite zobrazí – prejaví sa na výslednom obrázku. Vďaka takémuto experimentovaniu získavajú žiaci spätnú väzbu a cenné skúsenosti s tým, ako jednotlivé príkazy fungujú. Na predchádzajúcich dvoch príkladoch môžeme vidieť, že hranica medzi tvorbou algoritmov a programovaním sa stráca.

V súčasnosti, najmä pri vyučovaní programovania, považujeme za **dôležité**, aby žiaci čo možno najskôr zapisovali a overovali si algoritmy vo vhodnom prostredí na počítači.

V súčasnosti už **nepovažujeme** za potrebné učiť začiatočníkov, ako kedysi, rozmýšľať a zapisovať programy v špeciálnom „algoritmickom“ jazyku:

- Takýto jazyk mával často textovú podobu. Pritom vychádzal z prirodzeného jazyka s rôznymi frustrujúcimi dodatočnými obmedzeniami. Napríklad, pri hľadaní najnižšej paličky metódou „pozriem sa a vyberiem najnižšiu“ učiteľ zakáže operáciu „pozretia“ z pre žiakov nejasných dôvodov.

- Zvykli sa používať aj rôzne grafické zápisy, napríklad vývojové diagramy alebo štruktúrogramy. Pri ich zápise už bolo potrebné dodržiavať určité formálne pravidlá. Preto sa do určitej miery dajú tieto zápisy chápať ako programy, ktoré však nemáme šancu spustiť, vyskúšať.

Takéto prístupy považujeme za prekonané. Algoritmické jazyky tvoria zbytočnú medzivrstvu medzi nápadom, programovaním a výsledkom. V konečnom dôsledku tak oddaľujú etapu zbierania skúseností.

Programovanie v osnovách

Téma algoritmy a programovanie sa začína vyučovať už na základnej škole. Ďalej pokračuje na strednej škole. V jednotlivých stupňov vzdelávania sa riešia rôzne, a pritom veku primerané témy v rôznych rozsahoch.

Aktivita 4.2	Uvažujte nad otázkou: „Prečo je programovanie najdôležitejšou časťou informatiky na ZŠ, SŠ?“ Dôvody povedzte.

Obsah informatiky a informatickej výchovy je na každom stupni rozdelený na päť tematických okruhov (vychádzame z dostupných materiálov Štátneho pedagogického ústavu):

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

Ďalej sa zameriame sa na líniu *Postupy, riešenie problémov, algoritmické myslenie*:

- 1. stupeň ZŠ: Informatická výchova
- 2. stupeň ZŠ: Informatika
- Stredné školy: Informatika
- Stredné školy: zameranie na informatiku

Je dôležité pripomenúť, že informatika spolu s matematikou tvoria vzdelávaciu oblasť **Matematika a práca s informáciami**.

ISCED1 – 1. stupeň ZŠ: Informatická výchova

Pojmy:

- postup, návod, recept,
- riadenie robota, obrázková stavebnica, postupnosť krokov,
- detský programovací jazyk, elementárne príkazy, program,
- robotická stavebnica.

Vlastnosti a vzťahy, postupy a metódy:

- skladanie podľa návodov (stavebnice, hlavolamy, origami)
- zápis/vytvorenie postupu, receptu, návodu a práca podľa návodu
- v počítačovom prostredí riešenie úloh pomocou robota, skladanie obrázkov z menších obrazcov, okamžité vykonávanie príkazov, vykonanie pripravenej postupnosti príkazov
- riešenie jednoduchých algoritmov v detskom programovacom prostredí (kreslenie obrázkov, pohyb animovaných obrázkov)

Obsahový štandard:

- získanie základov algoritmického myslenia.

Výkonový štandard:

- získať základy algoritmického myslenia – príkazy v priamom režime,
- riešiť jednoduché algoritmy v detskom programovacom prostredí.

Komentár

Na prvom stupni hrá veľmi dôležitú úlohu vhodné prostredie, akými sú napríklad prostredia Cirkus šaša Tomáša, Baltík, Panák alebo Karel. Tiež sa predpokladá, že pre riešenie rôznych hlavolamov, bludísk alebo úloh, v ktorých je potrebné hľadať závislosti alebo dodržiavať stanovené pravidlá, vzniknú vhodné prostredia.

ISCED2 – 2. stupeň ZŠ: Informatika

Pojmy:

- postup riešenia, etapy riešenia problémov,
- programovací jazyk, elementárny príkaz, postupnosť, procedúra, cyklus,
- zložitosť riešenia problému.

Vlastnosti a vzťahy, postupy a metódy:

- v detskom programovacom prostredí riešenie úloh s opakovaním nejakých činností, zoskupovanie častí riešenia do procedúr,
- porovnanie času trvania rôznych riešení problému.

Obsahový štandard:

- postup riešenia, formálny zápis riešenia, etapy riešenia problémov,
- programovací jazyk, elementárny príkaz, postupnosť, cyklus, procedúra, parametre, premenná, hodnota, priradenie,
- zložitosť riešenia problému.

Výkonový štandard:

- žiak dokáže zapisovať a interpretovať postupy (napríklad zápis matematických algoritmov, algoritmus na jednoduché zašifrovanie textu),
- žiak demonštruje v detskom programovacom prostredí riešenie úloh s opakovaním nejakých činností, zapamätávanie výpočtov do premenných, zoskupovanie častí riešenia do procedúr,
- žiak dokáže porovnať čas trvania rôznych riešení problému.

Komentár

Aj na druhom stupni základnej školy bude potrebné venovať pozornosť programovaciemu prostrediu a programovaciemu jazyku. Odporúčané prostredia sú Imagine, Scratch, Karel, Baltík, prípadne detské verzie jazykov (napríklad Pascal).

Ak vyžadujeme formálne (textové) zápisy programov, je výhodné používať

interpretovaný jazyky v prostredí, ktoré umožňuje grafický výstup (Imagine–Logo). Žiaci potom získavajú prvé skúsenosti priamym zadávaním jednoduchých príkazov, ktoré čosi kreslia.

ISCED3 – stredná škola: Informatika

Pojmy:

- algoritmus, program, programovanie, etapy riešenia problému,
- príkazy (priradenie, vstup, výstup),
- riadiace štruktúry (podmienené príkazy, cykly),
- premenné, typy, množina operácií.

Obsahový štandard:

- 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.

Výkonový 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 použitia príkazov, premenných, typov a operácií,
- používať základné typy údajov v používanom programovacom jazyku,
- rozpoznať a odstrániť syntaktické chyby, opraviť chyby vzniknuté počas behu programu, identifikovať miesta programu, na ktorých môže dôjsť k chybám počas behu programu.

Komentár

Tematické plány môžu byť pre školy rôzne. Napríklad, v jednom návrhu tematických plánov sa programovaniu venuje takéto množstvo hodín:

- v 1. ročníku 0 hodín z 32
- v 2. ročníku 8 hodín z 32
- v 3. ročníku 16 hodín z 30

Téme *Postupy, riešenie problémov, algoritmické myslenie* sa venuje približne štvrtina celkového počtu hodín Informatiky. Napríklad, v inom návrhu pre školy so zameraním na informatiku má programovanie:

- v 1. ročníku 16 hodín (problém, algoritmus, etapy riešenia problému, programovací jazyk)
- v 2. ročníku 16 hodín (príkaz priradenia, vstup, výstup, riadiace podmienené príkazy, cykly, premenné, typy, množina operácií)
- v 3. ročníku 0 hodín

Programovanie spočiatku tvorí malú časť informatiky. Postupne však jeho význam narastá. Predpokladá sa, že maturant si vyberie voliteľný predmet tak, aby maturitu z informatiky úspešne zvládol. Pre maturity je programovanie veľmi dôležité. Ako sme mohli vidieť, programátorské úlohy na maturitách tvoria až 50% všetkých úloh.

Chceli by sme dať do pozornosti aj to, že vzdelávací program nepredpisuje žiadne programovacie jazyky ani prostredia. Vzhľadom na dobré skúsenosti a dostupnosť pedagogických materiálov by sme však odporučili požívať Imagine, Pascal, Delphi prípadne Lazarus. Pripomeňme ešte, že maturitným jazykom je Pascal.

Kapitola 5: O poznávacom procese

V tejto kapitole si povieme viac o tom, ako v súčasnosti rozumieme poznávaciemu procesu v informatike a najmä pri vyučovaní algoritmov a programovania. Považujeme za dôležité uvažovať o poznávacom procese preto, lebo sami chceme porozumieť tomu, ako prebieha pochopenie informatického poznatku. Dostávame sa tak na hranicu medzi informatikou a psychológiou.

Aktivita 5.1	Diskutujte: Je programovanie ťažké? Ak áno, prečo je to tak?

Rozdiel medzi tým, čo učíme a ako tomu žiaci porozumejú

Je relatívne, čo považujeme za jednoduché my, učitelia a čo žiaci.

Nezriedka sa stáva, že mladý učiteľ je zanietený, nadšený, entuziastický a má sklon učiť všetko, čo sám vie. Napríklad, učí žiakov tému polia tak, že zadeklaruje pole a ako ukážkový príklad predvedie triedenie. Môže tak nadobudnúť dojem spokojnosti, pretože stihol veľa vecí porozprávať. Pritom však skombinoval dve náročné témy – polia a triedenia. Zabudol aj vysvetliť základné pojmy: čo je index, čo je prvok poľa. Navyše, tieto pojmy nestačí iba vysvetliť, ich použitie si musia žiaci natrénovať. Najlepšie na malých, jednoduchých príkladoch: použitie indexu ako konštanty, naplnenie poľa hodnotami, vypísanie obsahu poľa na obrazovku atď.

Spomínané pojmy a algoritmy učiteľ ovláda, má ich natrénované, pretože s nimi bežne a dlhé roky pracuje. Preto sa učiteľovi zdajú polia jednoduché. Aj triedenie sa mu zdá jednoduché. Pre žiakov sú toto dve nové a veľmi náročné témy. Navyše sú aj zle vysvetlené, takže iba málo žiakov má šancu výkladu učiteľa porozumieť.

Pojmy v informatike

V informatike používame veľa odborných termínov, neznámych pojmov. Niektoré z nich dokážeme vysvetliť populárne, prípadne sa im dokážeme vyhnúť. Potom aj laik má akú-takú šancu porozumieť diskusii dvoch zanietených informatikov.

V súvislosti s algoritmami a programovaním však používame, a teda potrebujeme, aby naši žiaci porozumeli pojmom, ktoré sú **abstraktné**. Takými sú napríklad už aj elementárne pojmy, ako premenná, cyklus, vetvenie alebo súbor, graf, prehľadávanie alebo aj model automatu, efektívnosť, zložitosť, vypočítateľnosť.

Preto nás obzvlášť musí zaujímať:

- ako tieto pojmy vysvetlíme žiakom – ako vznikajú v ich predstavách
- ako im žiaci porozumejú alebo neporozumejú (a prečo)
- ako môžeme diagnostikovať nesprávne pochopené poznatky, a prečo k takýmto deformáciám dochádza

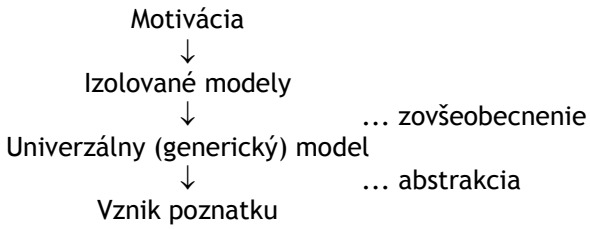
Za meradlo kvality vyučovania rozhodne nebudeme považovať to, koľko informácií učiteľ porozprával – teda, ako sa učiteľ pred žiakmi predvádza. Naopak, za kvalitné vyučovanie budeme považovať také, ktorému porozumie najviac žiakov.

Pri vyučovaní programovania máme dobré skúsenosti s aplikovaním teórie konštruktivizmu, princípov konštrukcionizmu a zážitkového učenia.

Aktivita 5.2	V čom spočíva zážitkové učenie?
Aktivita 5.3	Máte aj vy dobré skúsenosti so zážitkovým učením? Ak nie, prečo (v čom spočíva problém)?

Inšpirácia v matematike

V potrebe porozumieť abstraktným pojmom sa vyučovanie informatiky do istej miery podobá matematike. Preto sa oplatí pozrieť na to, ako matematici rozumejú poznávaciemu procesu. Existujú rôzne teórie o tom, ako vzniká poznatok. Nám je najbližšie teória, ktorú vypracoval prof. M. Hejný [8], [9]. Podľa neho vzniká matematický poznatok v etapách:



Napríklad zlomok, je abstraktný pojem:

- žiaci si ho najskôr predstavujú na rôznych izolovaných modeloch, napríklad:
  - polka chleba
  - štvrť masla
  - tretina torty
  - polovica čokolády
- modely im umožňujú získavať skúsenosti:
  - pomáhajú pochopiť zápisy (1/3 .. tretina torty)
  - umožňujú vykonávať operácie (+, –, *, /), relácie (=, ≠)
- nadobudnuté skúsenosti sa porovnávajú, zoskupujú a organizujú sa podľa spoločných vlastností
- postupne sa začne jeden model (niekedy aj viac modelov) častejšie používať – vzniká univerzálny model, pomocou ktorého sa ilustrujú operácie:
  - torta = kruh rozdelený na rovnaké mesiačiky,
  - čokoláda = obdĺžnik na štvorčekovom papieri
- univerzálne modely umožňujú vykonávať všeobecnejšie operácie, napríklad:
  - tretina čohosi + tretina čohosi = dve tretiny čohosi
- postupným organizovaním a preoganizovávaním skúseností vzniká schéma (všeobecný zápis, postup, princíp) až abstraktný poznatok.

Matematika je zatiaľ do veľkej miery odkázaná na modely a rôzne predstavy. Modely v matematike slúžia na to, aby žiaci pomocou nich získavali **skúsenosti**. Pre vyučovanie informatiky je takýto matematický pohľad veľmi cenný.

Totiž, aj v informatike používame v istých situáciách modely na to, aby sme vysvetlili nové pojmy. V informatike však nie sme na modely odkázaní do takej miery ako v matematike. Je to preto, lebo žiaci môžu získať praktické skúsenosti jednoduchšie – tým, že skúšajú veci na počítači.

Pri vyučovaní informatiky dávame prednosť takému učeniu, pri ktorom žiaci sami zbierajú **vlastné skúsenosti**. Teda napríklad aj tým, že žiaci algoritmy prakticky programujú. Oproti matematike tak získavame výhodu v tom, že žiaci si môžu algoritmy priamo vyskúšať, overiť, môžu sa s algoritmami „hrať“. Tým získajú ďalšie cenné skúsenosti. Preto sme si „zakázali“ učiť programovanie na papieri.

Z uvedeného vyplýva, že nie sme zástancami encyklopedických vedomostí (aj napriek tomu, alebo práve preto, že sa dobre skúšajú). Tie mávajú krátku trvácnosť a zbytočne zaťažujú žiakov faktami, ktoré málokedy využijú. Nie zriedka bývame svedkami toho, ako žiaci nedokážu aplikovať encyklopedické vedomosti na riešenie reálnych, a pritom triviálnych problémov. Pri ich riešení dokonca, ani netušia, že práve takéto svoje encyklopedické vedomosti majú aplikovať.

Vyučovanie programovania

Pri vyučovaní programovania sa vo všeobecnosti držíme nasledovnej schémy:

Motivácia
↓
Zbieranie skúseností
↓
Trénovanie poznatku

Jednotlivé etapy vysvetlíme na téme premenná. Pojem premenná je abstraktný pojem a s touto témou mávajú žiaci veľké problémy.

Predpokladané vedomosti

Ak začíname tému premenná učiť, predpokladáme, že žiaci už pracovali v programovacom jazyku. Dokážu zapísať postupnosť príkazov, ktorá čosi kreslí do grafickej plochy a rozumejú tomu, že v programe je dôležité poradie príkazov.

Aktivita 5.4	Povedzte, aké vedomosti predpokladáte u žiakov pred tým, ako budete preberať premenné.

Pokým pracovali v prostredí Delphi alebo Lazarus, už pravdepodobne nastavovali farbu textu alebo výplne. Preto už:

- vedia vypísať texty a čísla pomocou príkazu:
  `Image1.Canvas.TextOut(x, y, text);`
- videli príkaz priradenia, zatiaľ ho však používali v tvare
  `vlastnosť := hodnota`
- zoznámili sa s niektorými typmi údajov: `string`, `Integer`, `Real`, `TColor`
- dokážu zmeniť číslo na text a naopak
- vedia používať editovacie políčko na zadávanie vstupných údajov.

Príkaz `Image1.Canvas.TextOut` žiakom vysvetľujeme ako *príkaz na výpis textu*. Túto metaforu používame preto, aby sme sa vyhli nežiaducemu vysvetľovaniu objektov a volaniu metód. Pokým to nepreženieme, bývajú metafory výhodnou pomôckou, ako sa vyhneme vysvetľovaniu náročných konceptov (pozri aj modul Didaktika programovania 2).

Je výhodné, ak žiakov naučíme uvedené poznatky ešte pred tým, ako budeme učiť premenné (pozri aj [10], s. 5 až 11)). Premyslenou prípravou tak zamedzíme tomu, aby sme na jednej vyučovacej hodine naraz učili viacero nových vecí.

Analýza témy

Skúsme tému premenná analyzovať.

Aktivita 5.4	Povedzte, aké pojmy, syntaktické pravidlá a ďalšie vedomosti súvisia s premennými?

Vypíšme si pojmy, poznatky, syntaktické pravidlá, algoritmy, ktoré s premennou súvisia:

- samotný pojem premenná
- operácie: nastavenie hodnoty, zapamätanie hodnoty, použitie hodnoty
- syntaktické pravidlá: deklarácia, meno premennej, typ, príkaz priradenia
- algoritmy: priradenie, vypísanie hodnoty, použite hodnoty vo výrazoch, vyhodnotenie výrazu, načítanie vstupných údajov, výmena premenných
- neskôr aj:
  - lokálne a globálne premenné
  - záznamy, polia, objekty
  - dynamické premenné

Premenná má v programovaní iný význam ako v matematike. V matematike sa premenná chápe skôr ako neznáma hodnota, hľadaná hodnota. V programovaní je premenná synonymum pre pamäťové miesto.

Cieľ

Vidíme, že pri analýze sme objavili veľmi veľký počet nových poznatkov, ktoré nestihneme počas jednej vyučovacej hodiny naučiť tak, aby im žiaci rozumeli a aby ich dokázali použiť.

Stanovme si realistický vzdelávací cieľ: čo vieme žiakov počas jednej vyučovacej hodiny naučiť. Určite nebudeme na prvej hodine rozprávať všetko, čo o premenných sami vieme. Mnohé poznatky (ďalšie typy údajov, záznamy, polia) stačí, ak žiakov naučíme neskôr.

Cieľ pre jednu hodinu: naučiť používať jednoduché premenné (pracovať s pamäťou).

Motivácia

Motiváciou sa snažíme v žiakoch vyvolať túžbu spoznať nové poznatky, vyvolať vnútorný rozpor medzi tým, čo vedia a tým, čo chcú dosiahnuť. **Motivácia** je najdôležitejší faktor pri vyučovaní. Často sa však neuveriteľne podceňuje alebo úplne absentuje.

Vymyslieť dobrú motiváciu je najťažšia úloha učiteľa. Učitelia často roky skúšajú rôzne motivácie. Zatiaľ nevieme stanoviť všeobecné pravidlá pre to, ako motivovať žiakov. Je veľa faktorov, ktoré ovplyvňujú výber motivácie.

Veľmi záleží na tom, akú skupinu žiakov učíme: aké majú predchádzajúce skúsenosti, vek alebo sociálne postavenie. Motivácia, ktorá fungovala pri jednej skupine, už nemusí fungovať pre inú skupinu žiakov.

Motivácia je určená žiakom a má vyvolať v žiakoch potrebu, chuť naučiť sa nové poznatky. Preto nám pri hľadaní motivácie môže pomôcť, ak sa vžijeme do úlohy žiaka a položíme si otázky:

- „Prečo sa mám naučiť používať premennú?“
- „Načo mi to v živote bude?“

Motivácia môže byť založená na príbehu, situácii, úlohe, jasne formulovanom probléme, ktorý treba vyriešiť atď. Napríklad:

- „Chcem vytvoriť program, ktorý počíta stlačenia tlačidla.“
- „Do stredu grafickej plochy chcem nakresliť domček.“ Nakreslíme ho na tabuľu. Naznačíme súradnice. „Vidíme, že veľakrát počítame súradnice stredu `Image1.Width` **`div`** `2` **aj** `Image1.Height` **`div`** `2`. Pomohlo by nám, keby sme tieto hodnoty počítali iba raz a nejako si ich pamätali...“

Aktivita 5.5	Povedzte, akú motiváciu používate (by ste použili) pri vysvetľovaní premennej?

Od motivácie sa následne odvíja celá vyučovacia hodina: vzorový príklad, riešenie problému atď.

Preto má byť správna motivácia:

- pozitívna
- krátka, jednoduchá a stručná
- zrozumiteľná, vychádzajúca z existujúcich skúseností
- dostatočne silná

Pokým nás napadne viacero motivácií, zväčša volíme iba jednu z nich. V opačnom prípade sa stane vysvetľovanie nezrozumiteľné a žiaci nebudú rozumieť tomu „o čo učiteľovi ide“.

Príklady nevhodných motivácií:

- „Premenná sa deklaruje takto...“ ... toto s motiváciou nemá nič spoločné
- „Naučím vás premenné, pretože ich budete potrebovať.“ ... takýto sľub „svetlejších zajtrajškov“ je pre väčšinou žiakov slabou motiváciou
- „Vypočítajte korene rovnice $ax^2+bx+c=0$“ ... toto je sporná motivácia:
  - umelé matematické problémy nie sú v súčasnosti u žiakov obľúbené,
  - navyše riešenie matematického problému zakryje porozumenie pojmu premenná,
  - už v samotnom zadaní sa kombinujú premenné a parametre, takže žiakov takáto motivácia viac pomýli, ako pomôže pri vysvetľovaní.

Zbieranie skúseností

Ďalej zväčša potrebujeme vysvetliť niektoré nové idey, princíp alebo syntaktické pravidlá. Napríklad, musíme objasniť pojem premenná. Ak pracujeme v jazyku Pascal, potrebujeme vysvetliť aj jej deklaráciu.

Ukazuje sa, že samotné definície a teoretické vysvetľovanie nestačia. Podobne ako matematickým, tak aj informatickým pojmom žiaci najlepšie porozumejú tým, že nazbierajú dostatok vlastných **skúseností**.

Napríklad, predstavu o tom, čo je premenná, budujeme postupne. Prvé skúsenosti získajú žiaci už tým, že:

- sa pokúsime o krátke vysvetlenie: „Premenná je miesto v pamäti počítača.“
- využijeme škatuľkový model: na tabuľu nakreslíme miesto v pamäti počítača ako obdĺžnik (bez mena)
- použijeme prirovnanie: „Podobne, ako má kalkulačka jednoduchú pamäť M+, MR, aj počítač si dokáže zapamätať výsledky – počítač si však dokáže zapamätať veľmi veľa údajov.“

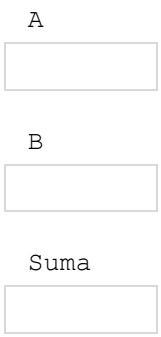
Predpokladáme pritom, že žiak už má určité skúsenosti z práce s kalkulačkou a s tým, že pracovali s počítačom (kreslili obrázky alebo písali texty, ktoré si počítač pamätal).

Pri programovaní sa nevyhneme vysvetľovaniu rôznych syntaktických pravidiel:

- „Ak chceme v programe používať premenné, musíme ich najskôr deklarovať:
  `var A, B Suma: Integer;`“
- Uvedený kúsok programu pritom píšeme na tabuľu, a tiež ukazujeme, na ktoré miesto v programe ho zapisujeme.
- Pravdepodobne budeme musieť upozorniť na medzery a symboly (`,` `:` `;`), ktoré pri zápise používame.

Pojem deklarácia je nový pojem, preto ho musíme vysvetliť:

- „Deklaráciou počítaču vysvetlíme, že chceme používať premenné – tie budú mať mená `A`, `B` a `Suma`“.
- Zároveň na tabuľu kreslíme škatuľky, ktoré majú uvedené pomenovanie.
- „Premenné `A`, `B` a `Suma` sú miesta v pamäti počítača. Premenná `A` je práve taká veľká, že si dokáže zapamätať celé číslo. Podobne, aj premenné `B` a `Suma` si dokážu zapamätať celé číslo – hodnotu typu Integer.“
- Gestom naznačíme, že slovo `Integer` vidíme v deklarácii.



Ďalej potrebujeme ukázať, akým spôsobom priraďujeme hodnoty do premenných a ako ich používame:

- „Premenné si dokážu pamätať hodnoty. Ak chceme, aby si premenná A pamätala číslo 1, použijeme príkaz priradenia:
  `A:=1;`“
- Opäť budeme musieť ukázať miesto v programe, kam príkaz zapisujeme.

A

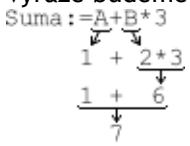
1

B

2

Suma

- „Čo sa stane, keď program vykoná predchádzajúci príkaz? Do škatuľky A sa uloží číslo `1`.“
- Do škatuľky `A` vpíšeme číslo 1.
- „Hovoríme, že do premennej `A` sme priradili číslo `1`.“
- „Hodnotu z premennej `A` vieme vypísať:
  `Image1.Canvas.TextOut(0, 0, IntToStr(A));`“
- „Do premennej `B` chceme priradiť číslo `2`. Viete povedať ako?“
- Očakávame zápis: `B:=2;`
- „Nakoniec vykonáme priradenie:
  `Suma:=A+B*2;`
  „Aká hodnota bude v premennej `Suma`?“
- Vo výraze budeme názorne trasovať priebeh výpočtu:



Ďalej sa môžu žiaci zamýšľať nad rôznymi malými problémami. Napríklad:

- „Aký bude obsah premenných, ak nakoniec ešte vykonáme príkaz `A:=10`?“
- „Aký bude obsah premennej `A` po priradení: `A:=A+1;`“

Takéto elementárne príklady, **malé kroky** považujeme za veľmi dôležité. Slúžia na to, aby žiaci:

- získali predstavu o premenných - preto kreslíme škatuľky
- zbierali skúsenosti s tým, ako sa do premenných nastavujú hodnoty
- zbierali skúsenosti s tým, ako sa z premenných čítajú hodnoty
- zbierali skúsenosti s vyhodnotením výrazu s premennými

Zbieranie skúseností je pre žiakov najcennejší, najhodnotnejší spôsob spoznávania, objavovania poznatkov.

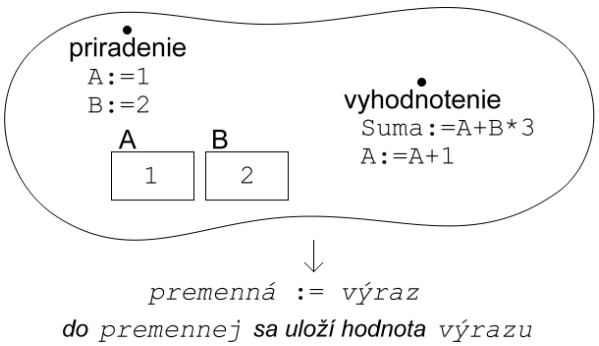
Aktivita 5.6	Čo si predstavujete pod zbieraním skúseností?

Zbieraním skúseností teda rozumieme:

- jednoduché vysvetlenia, prirovnania
- elementárne príklady, situácie, úlohy a mikroproblémy
- trasovanie príkazov

Tým, že žiaci získavajú skúsenosti, dostávajú šancu ich porovnávať, vytvárať medzi nimi vzťahy, vyvodzovať si vnútorné pravidlá, organizovať skúsenosti podľa spoločných a rozdielnych vlastností. Tým, že žiakom poskytujeme aj mierne odlišnejšie úlohy od tých, ktoré zažili doposiaľ (napríklad `A:=A+1`), sú nútení upraviť alebo preorganizovať svoju doterajšiu predstavu o pojme premenná.

Zoskupovanie funguje postupne ďalej podľa viac a viac všeobecných kritérií:



Nakoniec vidíme, ako sa zmenil jazyk, ktorým sa poznatok popisuje. Je úplne iný, ako bol na začiatku. Zápis „*premenná := výraz*“ je vysoko abstraktný. Je až na úrovni formálnej gramatiky, ktorou sa definuje syntax programovacieho jazyka. Aj opis operácie „*do premennej sa uloží hodnota výrazu*“ je vysoko abstraktný – v zápise sa už nevyskytujú konkrétne mená, ani konkrétne čísla, ani matematické operácie. Schopnosť porozumieť a používať takýto **metajazyk** je príznakom vysokej abstrakcie, ku ktorej sa žiaci dopracujú postupným zbieraním mnohých elementárnych skúseností.

Nový poznatok sa tak stáva súčasťou siete iných poznatkov. Samotný pojem premennej sa v budúcnosti ešte dopĺňa a formuje tak, ako sa žiaci postupne stretávajú s poľami, záznamami, dynamickými premennými alebo objektmi.

Trénovanie poznatku

Nový poznatok je potrebné trénovať a precvičovať. V informatike zvykneme poznatky často trénovať na gradovaných úlohách, ktoré žiaci samostatne riešia.

Napríklad:

- Nakreslite štvorec so stranou, ktorú zadal používateľ do editovacieho políčka
- Nakreslite krúžok s polomerom 10, ktorý má stred v náhodne zvolenom bode
- Povedzte, ktoré deklarácie sú správne, ktoré sú nesprávne a prečo:
```
var
  A, B: Integer;
  C: cislo;
  x, X, súčet: Integer;
```
- Aký bude obsah premenných po priradeniach:
```
var
  u, v: Integer;
begin
  u:=5;
  v:=u+10;
  u:=3;
  v:=v+u;
```
- Napíšte program, ktorý si od používateľa vypýta cenu jednej pesničky a počet pesničiek, ktoré plánujeme kúpiť. Potom vypočíta sumu, ktorú by sme za pesničky zaplatili.
- Vytvorte jednoduchú kalkulačku, ktorá sčítava čísla. Do editovacieho políčka zadáme číslo. Po stlačení tlačidla *Plus* sa číslo pripočíta k súčtu. Ak napríklad program spustíme a zadáme: `1` *Plus* `2` *Plus* `3` *Plus*, zobrazí sa výsledok `6`.
- Doplňte kalkulačku aj o ďalšie funkcie: zmazanie (vynulovanie) výpočtu, odčítanie, násobenie a delenie, rovná sa.
- Doplňte kalkulačku aj o funkcie M+, M-, MR a MC.

Aktivita 5.7	Aké ďalšie úlohy zvyknete riešiť?

Zhrnutie

V tejto kapitole sme sa pozreli na vyučovanie algoritmov a programovania z pohľadu poznávacieho procesu. Etapy poznávacieho procesu sme ilustrovali na konkrétnom príklade vyučovania premenných. Videli sme, že aj taká základná téma, akou sú premenné, obsahuje z pohľadu žiaka množstvo informácií a poznatkov.

Ak by sme žiakom porozprávali všetko, čo o premenných vieme, nášmu výkladu by nemali šancu porozumieť. Možno by sa fakty dokázali naučiť naspamäť a vedeli by ich na najbližšej hodine aj prerozprávať. Ich poznatky by však boli formálne – nerozumeli by pojmom, nevedeli by vyriešiť ani jednu úlohu.

Preto musíme byť pri vyučovaní programovania obzvlášť obozretní a musíme veľmi zvažovať, ktoré poznatky a akým spôsobom žiakov naučíme. Musíme sa do najmenších detailov zaujímať o to, čo žiakom povieme, čo kreslíme na tabuľu, aké úlohy riešime. Je napríklad neprípustné, aby sme uprostred hodiny začali používať nevysvetlené pojmy alebo príkazy. Často sa musíme vzdať aj rôznych (pre nás) krásnych grafických efektov alebo algoritmov. Nám sa síce javia ako úžasné, avšak pre splnenie cieľa vyučovacej hodiny sú nepodstatné, samoúčelné. Žiaci ich dokonca môžu chápať ako mágiu, ktorú im predvádzame, a ktorej nemajú šancu porozumieť.

Analýza témy nám má pomôcť v tom, aby sme si uvedomili, koľko pojmov, faktov alebo technických detailov sa k téme viaže. Takmer vždy ich objavíme obrovské množstvo. Z nich zvykneme vybrať ten najdôležitejší poznatok a možno zopár (1 až 3) faktov alebo technických detailov, ktorým sa nevyhneme. Ako pedagógovia a informatici, ktorí majú v porovnaní so žiakmi nad programovaním nadhľad, by sme mali byť schopní zvoliť si ten najdôležitejší cieľ, odkaz, **posolstvo**, ktoré chceme žiakov na danej hodine naučiť. Napríklad, pri premenných to nebudú všetky možné typy údajov, záznamy, polia ani dynamické premenné. Premenné sú v programovacom jazyku nástroj na to, aby sme si údaje dokázali zapamätať, spracovať, použiť, zobraziť. Na prvej hodine nám bude stačiť, ak žiaci dokážu do premennej priradiť, zmeniť jej hodnotu a vypísať ju na obrazovku.

Veľký dôraz kladieme na to, aby žiak postupne nazbieral veľa **vlastných skúseností**. Pri vyučovaní novej témy začíname po malých krokoch, ukazujeme jednoduché situácie, riešime krátke a triviálne príklady. Nechávame si dostatok času na to, aby sme so žiakmi tieto kroky absolvovali. Až potom postupne riešime gradované a mierne náročnejšie úlohy alebo problémy. Nezabudnime, že dobrá pozitívna motivácia hrá pri vysvetľovaní, zadávaní úloh aj riešení problémov veľmi dôležitú úlohu.

Kapitola 6: Programovanie v prostredí Imagine

V tejto kapitole sa venujeme vyučovaniu programovania v prostredí Imagine Logo. Pritom sa zameriame na druhý stupeň základnej školy.

Témy

Zvolili sme nasledujúce témy:

1. Úvod do prostredia Imagine Logo
2. Cyklus
3. Vlastné príkazy
4. Udalosti v živote korytnačky
5. Viac korytnačiek

Pri ich zostavovaní sme vychádzali z učebnice Tvorivá informatika – 1. zošit z programovania [2]. Zo skúseností vieme, že uvedeným témam porozumejú žiaci šiesteho ročníka. Vzhľadom na súčasné zmeny v osnovách by sme odporúčali až 7. ročník.

Aktivita 6

Zostavte 5 až 6 členné tímy.

Každý tím si zvolí jednu tému. Tému spracujte v tíme tak, ako by sa mala učiť na jednej 45 minútovej hodine. Pri príprave môžete čerpať námety z učebnice Tvorivá informatika – 1. zošit z programovania [2]. Nakoniec zástupca tímu prezentuje pri tabuli výsledok.

Prezentáciu začnete s analýzou tematického celku. Uveďte:

- zoznam predpokladov – čo už žiaci vedia
- cieľ – čo chceme naučiť
- zoznam pojmov a poznatkov, ktoré sa žiaci naučia

Ďalej predveďte, ako by mala vyzerať vzorová hodina. Teda, ako budete postupovať pri vyučovaní (čo budete vysvetľovať, akým spôsobom, čo budú riešiť žiaci). Je nevyhnutné, aby ste rešpektovali všetky kritéria poznávacieho procesu:

- treba začať vhodnou motiváciou
- na čom budú žiaci zbierať prvé skúsenosti (t.j. elementárne príklady, situácie, úlohy)
- aké ďalšie príklady alebo úlohy budú žiaci samostatne riešiť

Ostatní kolegovia na konci zhodnotia vystúpenie.

Môžete predpokladať, že výučba prebieha v počítačovej učebni, každý žiak sedí sám pri počítači, k dispozícii máte dataprojektor a tabuľu. Môžete predpokladať aj, že ste už zapísali veci do triednej knihy, prípadne zopakovali učivo z minulej hodiny.

Úvod do prostredia Imagine Logo

Predpoklady

Žiak:

- má základnú digitálnu gramotnosť, dokáže spustiť aplikáciu,
- pozná stupne z matematiky.

Schopnosť žiaka vyhľadať a načítať súbor využijeme v prostredí Imagine Logo.

Učiteľ:

- pred touto hodinou pracoval so žiakmi v grafickom editore, v ktorom si žiaci nakreslili obrázok krajinky s trávou a domčekom.

Cieľ

Príkazy

- `dopredu`
- `vľavo`
- `vpravo`

- spoznať programovacie prostredie Imagine Logo,
- naučiť sa ovládať korytnačku pomocou základných príkazov,

Nové pojmy a poznatky

Pojmy

- stránka
- korytnačka
- príkazový riadok
- vstup príkazu
- pozadie stránky
- pomôcka

- stránka a korytnačka v prostredí Imagine Logo, príkazový riadok,
- základné príkazy `dopredu`, `vľavo`, `vpravo` a ich vstupy (parametre),
- pomôcky pre základné príkazy `dopredu`, `vľavo`, `vpravo`,
- načítanie a uloženie pozadia stránky.

Motivácia

Postupy

- vykonať príkaz
- načítať pozadie
- uložiť pozadie
- otvoriť pomôcku pre príkaz

Učiteľ začne hodinu úvodnou motiváciou – porozpráva o korytnačke, ktorá žije na piesku a keď chodí po tomto piesku, zanecháva na ňom svoje stopy. Takúto korytnačku máme aj na počítači a dokážeme ju riadiť niekoľkými príkazmi.

Zbieranie skúseností

Učiteľ vysvetlí, predvedie a ukáže, ako funguje základný príkaz `dopredu` tým, že do **príkazového riadku** napíše:

`? dopredu 100` a stlačí kláves **Enter**

Učiteľ vysvetlí, že za príkazom je potrebné zapísať medzeru a potom počet krokov, o ktorý sa má korytnačka pohnúť. Na vykonanie príkazu musíme stlačiť kláves **Enter**.

Najčastejšie chyby žiakov – medzi príkaz a jeho vstup nevložia medzeru.

Žiaci vyskúšajú tento príkaz na svojich počítačoch.

Učiteľ podobne ako príkaz `dopredu` vysvetlí aj príkazy `vľavo`, `vpravo`. Tiež spomenie, že základné príkazy majú svoje skrátené tvary – `do`, `vl` a `vp`.

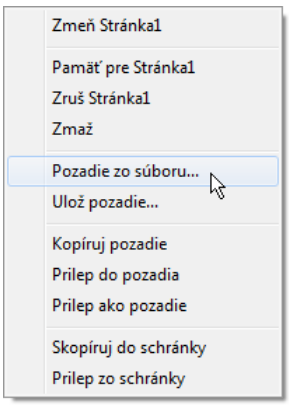
Žiaci skúšajú aj tieto príkazy a riadia pomocou nich korytnačku.

Učiteľ ukáže možnosť využívať kláves **šípka hore**, ktorá umožňuje nájsť príkaz, ktorý sme už niekedy zapísali do príkazového riadku. Po nájdení príkazu a stlačením klávesu **Enter** korytnačka tento príkaz znovu vykoná. Do príkazového riadku zapíše:

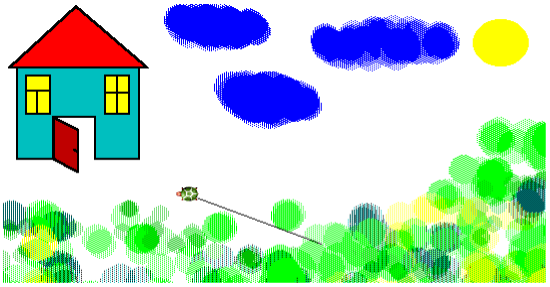
`? do 50` a stlačí **Enter**

Potom stlačí šípku hore. Upozorní, že sa v príkazovom riadku znovu objavil predtým napísaný príkaz `do 50`. Keď znovu stlačí **Enter** korytnačka sa znovu pohne.

Žiaci si vyskúšajú pohyb korytnačky pomocou zopakovania niektorého z doteraz zapísaných príkazov.



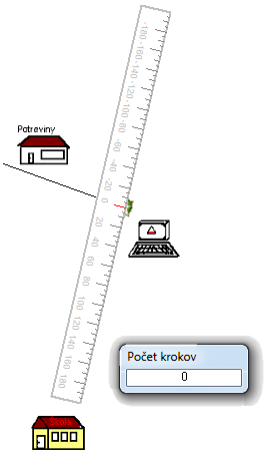
Učiteľ ďalej spomenie, že by mohlo byť zaujímavé, keby korytnačka nežila na bielej stránke, ale napr. v nejakej krajinke. Ukáže, ako do pozadia stránky načítať obrázok. Kliknutím pravého tlačidla myši do stránky a výberom položky **Pozadie zo súboru** otvorí dialógové okno, v ktorom nájde a vyberie súbor s obrázkom. Ukončí dialóg tlačidlom **Otvor**. Do pozadia stránky sa vloží vybratý obrázok.



Nájsť obrázok v dialógu, ktorý sa otvorí je rovnaké ako pri otváraní obrázka v grafickom editore.

Niektorí žiaci môžu mať problémy s otáčaním korytnačky vľavo a vpravo.

Žiaci na svojich počítačoch načítajú do pozadia stránky obrázok, ktorý si predtým pripravili na inej hodine. Snažia sa dostať korytnačku na určené miesto – do domčeka.



Učiteľ sleduje prácu žiakov na hodine, usmerňuje ju a riadi tak, aby žiaci pochopili základné príkazy a prácu s nimi.

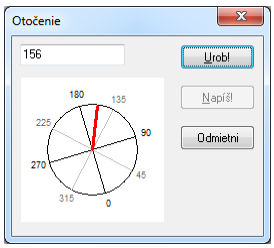
Učiteľ zopakuje, ako načítať do pozadia stránky obrázok tým, že do nej načíta nový obrázok, v ktorom je viac miest, kam sa má korytnačka dostať. Navrhne, aby korytnačka postupne prešla na všetky miesta. Niekoľko krokov ukáže tak, že bude korytnačku riadiť s využívaním pomôcok pre príkazy. Zapíše:

```
? do
```
a stlačí **Enter**

Objaví sa pravítko, v ktorom vyberie počet krokov. Pomôcka sa kliknutím do pravítka zatvorí. V príkazovom riadku sa objaví počet krokov a príkaz sa vykoná, t.j. korytnačka prejde na nové miesto na stránke.

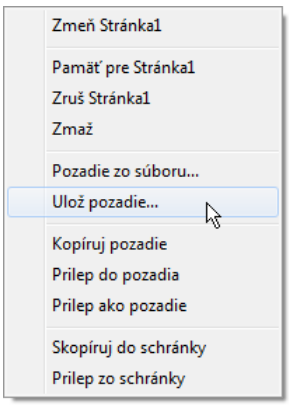
Potom učiteľ zapíše:

```
? vl
```
a stlačí **Enter**



Na stránke sa objaví kompas, v ktorom vidieť súčasné natočenie korytnačky. Učiteľ vyberie nové natočenie a stlačí tlačidlo **Urob**. Upozorní žiakov na to, čo sa udialo v príkazovom riadku – za príkaz sa dopísal počet stupňov, o ktoré sa má korytnačka otočiť.

Žiaci rovnakým spôsobom pracujú so svojimi obrázkami – načítajú si obrázok do pozadia a snažia sa dostať korytnačku na určené miesta. Takto získavajú veľa skúseností so základnými príkazmi `dopredu`, `vľavo`, `vpravo` a s využívaním pomôcok ku nim.



Učiteľ ukáže postup na uloženie pozadia – tento postup je obdobný ako načítanie obrázka do pozadia stránky. Žiaci si uložia obrázky s prechádzkami.

Učiteľ ukončí hodinu zhrnutím nových príkazov a ich využívaní pri riadení korytnačky ma stránke.

Trénovanie poznatkov

Na ďalších hodinách riešia žiaci úlohy z učebnice [2] zo strany 7 až 11.

Cyklus

Predpoklady

Žiak

- pozná prostredie Imagine Logo,
- v priamom režime dokáže ovládať korytnačku pomocou základných príkazov **dopredu**, **vpravo**, **vľavo**,
- dokáže robiť nastavenia pre pero pomocou základných príkazov **nechHP**, **nechFP** a kresliť farebné body pomocou základného príkazu **bod**,
- dokáže do stránky vložiť tlačidlo a do jeho udalosti **priZapnutí** napísať príkazy.

Príkazy na nastavenie hrúbky, farby pera a príkaz **bod** sa žiaci naučili riešením úloh z učebnice. Rovnako aj postup ako vložiť do stránky tlačidlo a definovať jeho udalosť **priZapnutí**.

Cieľ

- **pochopiť** základnú konštrukciu príkazu cyklu s pevným počtom opakovaní,
- v obrázkoch **vyhľadať** a **rozpoznať** opakujúce sa časti,
- **navrhnúť** príkazy, ktoré je potrebné opakovať, aby sa nakreslil obrázok,
- vytvoriť **konštrukciu cyklu opakuj** v programovacom jazyku.

Príkazy

- **opakuj**

Pojmy

- cyklus

Postupy

- vyhľadať opakujúcu sa časť v obrázku
- zápis cyklu **opakuj** v programovacom jazyku

Nové pojmy a poznatky

- cyklus **opakuj**,
- **zápis** cyklu **opakuj** v programovacom jazyku Imagine Logo.

Motivácia

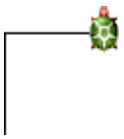
Učiteľ do stránky načíta pozadie, na ktorom bude ulica s domčekom, pri ktorom budú chýbať schody. Navrhne žiakom, aby tieto schody dokreslili.

Zbieranie skúseností

Keďže ide o veľmi dôležitú konštrukciu je nevyhnutné, aby si žiaci osvojili tento koncept správne.

Učiteľ spolu so žiakmi postupne zostaví príkazy, ktoré kreslia jeden schod:

```
? do 50
? vp 90
? do 50
? vl 90
```



Najčastejšia chyba – žiak nenájde celú opakujúcu sa postupnosť. Pripraví riešenie úlohy tak, že pred začiatkom cyklu aj po jeho ukončení zapíše príkazy, ktoré by mohli už patriť do vnútra cyklu.

Učiteľ vysvetlí, že je dobre, ak korytnačka skončí natočená tak ako začala, aby po nakreslení jedného schodu bola pripravená kresliť nasledujúci schod.

Učiteľ nakreslí druhý schod tak, že príkazy potrebné na nakreslenie schodu zapíše do jedného riadku:

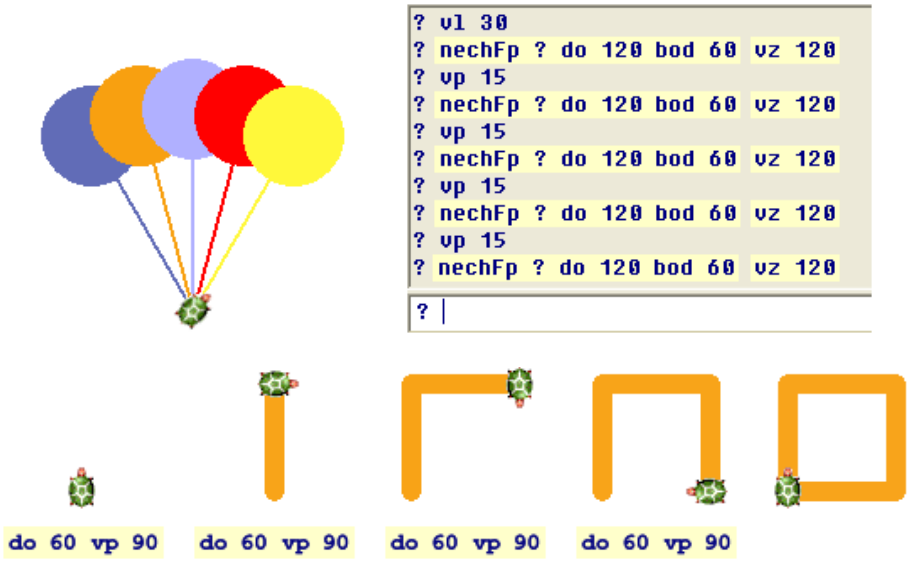
```
? do 50 vp 90 do 50 vl 90
```

Učiteľ nakreslí tretí schod tak, že predtým napísané príkazy dostane do príkazového riadka pomocou klávesu **šípka hore** a stlačením klávesu **Enter** ich korytnačka vykoná. Učiteľ postup ešte raz predvedie a povie, že takto sme našli postupnosť príkazov, ktoré sú potrebné na nakreslenie jedného schodu.

Žiaci sa snažia dokresliť schody na svojom obrázku.

Učiteľ nabáda žiakov, aby do jedného riadku pripravili niekoľko príkazov, ktoré je potrebné opakovať, a potom použili kláves **šípka hore**, aby sa tieto príkazy objavili v príkazovom riadku a aby ich tak mohli opakovane vykonávať.

Učiteľ ďalej využije učebnicu [2] a podľa strany 12 vysvetlí kreslenie balónikov a štvorca.



Žiaci si pri kreslení schodov, balónikov a štvorca uvedomujú, že pri kreslení obrázkov má zmysel vyhľadávať postupnosti príkazov, ktoré sa budú opakovať, čo im často uľahčí prácu pri kreslení zložitejších obrázkov.

Keďže pracujeme v priamom režime, žiaci si môžu každý svoj obrázok na pozadí uložiť zvlášť do súboru.

Učiteľ vysvetlí žiakom, že ak potrebujeme opakovať nejakú postupnosť krokov, rozmyslíme si, aká je táto postupnosť a potom využijeme konštrukciu cyklu `opakuj`. Učiteľ vysvetlí jednotlivé časti zápisu cyklu `opakuj` v programovacom jazyku Imagine Logo podľa učebnice [2]:

Učiteľ navrhne žiakom, aby vyhľadávali opakujúce sa postupnosti príkazov aj v ďalších obrázkoch, a potom tieto použili v cykle `opakuj` a obrázky nakreslili.

Trénovanie poznatkov

Žiaci vyhľadávajú opakujúce sa postupnosti príkazov pri kreslení obrázkov a využitím príkazu `opakuj` kreslia nasledujúce obrázky:

Žiaci z učebnice [2] zo str. 13 a 14 riešia úlohy 1 až 6.

Ak je to možné, učiteľ zobrazí obrázok pomocou dataprojektora a spolu so žiakmi uvažuje, či a kde sa v obrázku nachádzajú opakujúce sa časti. Tiež môže zobraziť riešenie niektorého žiaka pomocou dataprojektora a zdôrazňuje v ňom využívanie príkazu `opakuj`.

Žiaci si môžu svoje obrázky priebežne ukladať.

Učiteľ na konci zhrnie vyučovaciu hodinu, na ktorej sa využíval k cyklus `opakuj`.

Vlastné príkazy

Predpoklady

Žiak

- pozná prostredie Imagine Logo,
- pracoval v priamom režime, v ktorom riadil korytnačku pomocou základných príkazov `dopredu`, `vľavo`, `vpravo`,
- dokáže kresliť obrázky, v ktorých mení nastavenia farby a hrúbky pera pomocou základných príkazov `nechHP`, `nechFP` a ich vstupov,
- používa základný príkaz `bod`, s konkrétnym aj náhodným vstupom na kreslenie bodov,
- v obrázkoch dokáže vyhľadávať opakujúce sa postupnosti príkazov a navrhovať ich kreslenie pomocou príkazu `opakuj`.

Príkazy

- `uprav`

Pojmy

- vlastný príkaz,
- editor príkazov,

Postupy

- vytvoriť vlastný príkaz,
- upraviť vlastný príkaz,
- otvoriť projekt,
- uložiť projekt

Cieľ

- rozumieť, že sa korytnačka dokáže naučiť aj nové príkazy,
- vedieť vytvárať vlastné príkazy,
- dokázať upravovať vlastné príkazy,
- vyvolať vlastné príkazy,
- vedieť uložiť a otvoriť projekt v prostredí Imagine Logo.

Nové pojmy a poznatky

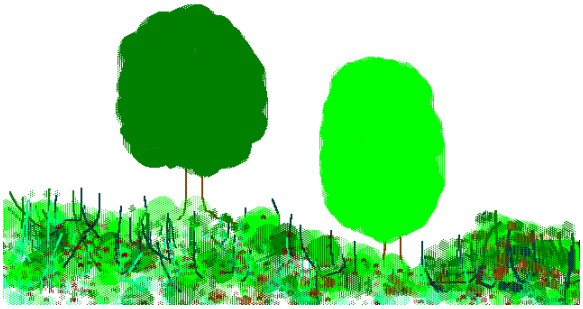
- základný príkaz `uprav`,
- editor príkazov – otvorenie, ukončenie, prezeranie a úpravy vlastných príkazov,
- uložiť projekt,
- načítať projekt.

Motivácia

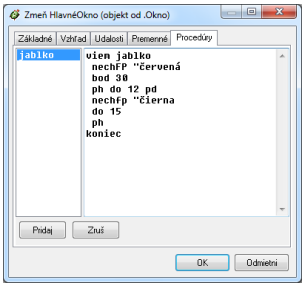
Učiteľ zopakuje predchádzajúce poznatky tým, že v priamom režime nakreslí obrázok, napr. niektorý z učebnice [2] na str. 14:

```
? vp 90 nechFP "purpurová opakuj 15 [bod 10 ph do 10 pd]
```

Učiteľ do stránky načíta obrázok sadu, v ktorom je potrebné dokresliť na jeden strom jablká a na druhý hrušky.



Žiaci budú zrejme príkazy na nakreslenie jablka zapisovať postupne do viacerých riadkov.



Žiaci na zopakovanie základných príkazov a práce v priamom režime nakreslia jedno jablko – červený bod a čiernu čiarku:

```
? nechfp "červená bod 30 ph do 12 nechfp "čierna pd do 10
```

Učiteľ porozpráva o tom, že takýchto jabĺk budeme potrebovať nakresliť veľa, aby

bola bohatá úroda. Predvedie, že korytnačke je po dokreslení jablka potrebné zdvihnúť pero a premiestniť na iné miesto na strome, a potom znovu vyhľadať alebo napísať všetky príkazy na kreslenie jablka. Takéto vyhľadanie všetkých príkazov v správnom poradí nemusí byť jednoduché, ale jablko by sa dalo kresliť oveľa jednoduchšie, keby to bolo jedno slovo, ktoré by korytnačka už poznala.

Zbieranie skúseností

Učiteľ ukáže otvorenie editora vlastných príkazov pre príkaz `jablko` tým, že zapíše:

```
? uprav "jablko
```

Učiteľ v editore napíše príkazy na kreslenie jablka. Editor zatvorí tlačidlom **OK**.

Učiteľ ukáže použitie nového príkazu tým, že ho zapíše do príkazového riadku:

```
? jablko
```

Učiteľ vysvetlí, že od tejto chvíle korytnačka rozumie príkazu `jablko` a vždy nakreslí takýto obrázok. Presunie korytnačku ešte na ďalšie miesto a nakreslí jablko.

Učiteľ navrhne, že možno bude lepšie vyzerať hnedá stopka. Tým, že znovu otvorí editor

```
? uprav "jablko
```

upraví farbu stopky, ukončí editor, presunie korytnačku a vyskúša opravený príkaz zapísaním:

```
? jablko
```

Žiaci navrhnú vlastný príkaz na kreslenie jablka. Vyskúšajú si jeho funkčnosť na prázdnej stránke tým, že presúvajú korytnačku a nakreslia niekoľko jabĺk. Keď sa pomýlia pri definovaní príkazu alebo zabudnú po jeho dokreslení zdvihnúť pero, upravia príkaz tým, že ho znovu otvoria v editore.

Žiaci načítajú do stránky pozadie sadu, presúvajú korytnačku po korunách stromov a kreslia jablká. Priebežne si ukladajú obrázok pozadia do súboru – rovnako ako na minulých hodinách.

Učiteľ zadá žiakom úlohu, aby vytvorili vlastný príkaz `hruška`. Pomocou neho treba nakresliť niekoľko hrušiek na druhý strom.

Učiteľ ukáže, ako uložiť projekt – v hlavnej ponuke v položke **Súbor** výberom **Uložiť projekt ako...**

Žiaci si uložia svoje projekty.

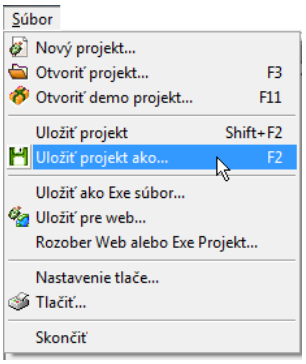
Učiteľ ukáže, ako začať nový projekt – v hlavnej ponuke v položke **Súbor** výberom **Nový projekt**. Tým, že navrhne vlastný príkaz `balón` zopakuje možnosť vytvárania nových príkazov.

Žiaci si vytvoria nový projekt a pripravujú svoje vlastné príkazy napr. štvorec a ďalej také, ktoré riešia ukážkové úlohy z učebnice [2] na stranách 16.

Učiteľ zhrnie nové pojmy a poznatky z hodiny.

Čítanie obrázka do pozadia žiaci robili na minulých hodinách.

Postupnosť príkazov na kreslenie balóna už žiaci videli, keď ich využívali v priamom režime. Nové v tejto aktivite je to, že vytvoria vlastný príkaz, ktorý kreslí balón.



Trénovanie poznatkov

Na niekoľkých nasledujúcich hodinách si žiaci precvičujú nadobudnuté vedomosti riešením úloh z učebnice [2] na stranách 16 až 18.

Udalosti v živote korytnačky

Predpoklady

Žiak

- pozná prostredie Imagine Logo,
- je schopný v priamom režime riadiť korytnačku pomocou základných príkazov,
- vie kresliť obrázky, v ktorých menil nastavenia farby a hrúbky pera a využíval základný príkaz `bod`,
- dokáže vytvárať, upravovať a vykonávať vlastné príkazy,
- je schopný načítať a uložiť projekt v prostredí Imagine Logo.

Cieľ

Príkazy

- `nechPoz`

Pojmy

- rodný list,
- udalosť,
- definovanie udalosti

Postupy

- otvoriť rodný list korytnačky,
- pridať udalosť,
- zrušiť udalosť,
- zatvoriť rodný list korytnačky

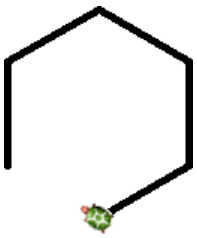
- definovať reakciu na udalosť **priKliknutí**,
- zrušiť udalosť v rodnom liste korytnačky,
- klikať na korytnačku a kresliť takto rôzne obrázky.

Nové pojmy a poznatky

- rodný list korytnačky,
- udalosť **priKliknutí**,
- definovanie príkazov pre udalosť,
- základný príkaz `nechPoz` a jeho náhodný vstup.

Motivácia

Učiteľ povie, že na korytnačku môžeme klikať myšou a môžeme ju naučiť, aby na takéto kliknutie reagovala. Aj takýmto spôsobom môže niečo nakresliť.



Zbieranie skúseností

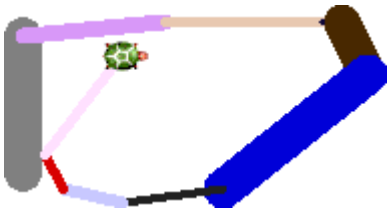
Učiteľ podľa strany 20 v učebnici [2] otvorí rodný list korytnačky, nájde v ňom miesto, kam sa zapisujú príkazy, ktoré korytnačka vykoná pri kliknutí. Zapíše tam príkazy:

```
priKliknutí: do 70 vp 60
```

Učiteľ predvedie, čo sa stane, keď rodný list zatvorí a klikne na korytnačku. Keď dokreslí celý šesťuholník, znovu otvorí rodný list a upraví príkazy pre udalosť **priKliknutí**, napr.

```
priKliknutí: nechFP ? nechHP ? do ? vp ?
```

Učiteľ znovu zatvorí rodný list a klikaním na korytnačku predvedie kreslenie čiar náhodnej farby, hrúbky a dĺžky.



Žiaci definujú príkazy pre udalosť **priKliknutí** a klikaním na korytnačku kreslia čiary.

Motivácia

Učiteľ povie, že korytnačka dokáže v ploche skákať na rôzne miesta.

Zbieranie skúseností

Učiteľ podľa strany 21 v učebnici [2] ukáže základný príkaz `nechPoz` zapísaním do príkazového riadku:

```
? nechPoz ?
```

Vysvetlí správanie korytnačky pri jeho vykonaní. Navrhne, aby korytnačka tento príkaz vykonávala vo svojej udalosti **priKliknutí**.

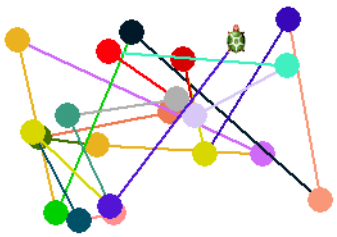
Trénovanie poznatkov

Učiteľ ďalej navrhne, aby žiaci vymysleli, aké obrázky by takto mohli nakresliť na stránku.

Žiaci definujú svoje vlastné obrázky a vkladajú ich do udalosti **priKliknutí**. Môžu to byť bodky, hviezdičky, domčeky, atď.

Učiteľ pomáha žiakom pri práci s novými poznatkami – otváranie a práca s rodným listom, návrhom a skúšaním udalosti **priKliknutí**. Ak má možnosť, zaujímavé obrázky premietne cez dataprojektor.

Učiteľ zadá úlohu naprogramovať korytnačku tak, aby pri kliknutí skočila na náhodné miesto a nakreslila bod náhodnej farby.



Žiaci riešia úlohu.

Žiaci ďalej zo strany 21 v učebnici [2] riešia úlohu 4.

Učiteľ zosumarizuje priebeh hodiny, spomenie nové pojmy a poznatky, ktoré sa žiaci na nej naučili.

Čo sme sa naučili v tomto module

Zhrnutie

Poznáme postavenie programovania v rámci informatiky na základnej a strednej škole.

Dozvedeli sme sa a vyskúšali sme si niekoľko detských prostredí, ktoré sú vhodné na vyučovanie programovania na základnej škole.

Pre jednotlivé programátorské témy základnej školy dokážeme navrhnúť didakticky správnu postupnosť motivácií, príkladov a úloh.

Preverenie výstupných vedomostí

Zvoľte si inú tému z programovania v Logu alebo ľubovoľnú tému z programovania v prostredí Scratch, Baltík. Analyzujte ju a zostavte metodický materiál pre jednu vyučovaciu hodinu tak, aby rešpektoval spomínané didaktické pravidlá.

Literatúra a použité zdroje

[1] Blaho, A. (2006) Informatika pre stredné školy. Programovanie v Delphi. Bratislava: SPN. ISBN 80–10–00421–9
[2] Blaho, A., Kalaš, I.: Tvorivá informatika. 1. zošit z programovania. Bratislava : SPN – Mladé letá, 2007. 48 s. ISBN 80-10-01223-7
[3] Bezáková D., Lovászová G., Kučera, P.: Programovanie 1. Bratislava: ŠPÚ. ISBN 978-80-89225-65-1
[4] Bezáková D., Kučera, P., Lovászová G., Tomcsányi, P.: Programovanie 4 (Logo). Bratislava: ŠPÚ. ISBN 978-80-8118-017-0
[5] Blaho, A., Kubincová, Z., Salanci, Ľ. (2009) DVUI: Programovanie 2. Bratislava: ŠPÚ. ISBN 978–80–8118–007–1
[6] Blaho, A., Kubincová, Z., Salanci, Ľ. (2009) DVUI: Programovanie 3. Bratislava: ŠPÚ. ISBN 978–80–8118–014–9
[7] Blaho, A., Kubincová, Z., Salanci, Ľ. (2009) DVUI: Programovanie 4 (Pascal). Bratislava: ŠPÚ. ISBN 978-80-8118-018-7
[8] Hejný, M. a kol: Teória vyučovania matematiky. Bratislava: SPN 1990. ISBN 80–08–013443–3
[9] Hejny, M.: Understanding and structure, Proc. of CERME 3. Bellaria 2003. www.dm.unipi.it/~didattica/CERME3/proceedings/Groups/TG3/TG3_Hejny_cerme3.pdf
[10] Salanci, Ľ.Blaho, A., Kubincová, Z. *DVUI: Programovanie 2*. Bratislava: ŠPÚ 2009. ISBN 978–80–8118–007–1

Webové stránky:

[11] http://www.statpedu.sk/buxus/docs//Pedagogicke_dokumenty/zakladne_skoly/osnovy/Informaticka_vychova_1st_uo.pdf
[12] http://www.statpedu.sk/buxus/docs//Pedagogicke_dokumenty/zakladne_skoly/osnovy/Informatika_2st_uo.pdf
[13] http://www.statpedu.sk/Pedagogicke_dokumenty/Gymnazia/8roc/Osnovy/INFORMAT.DOC
[14] http://www.statpedu.sk/buxus/docs//Pedagogicke_dokumenty/Gymnazia/8roc/plan/UP_8_roc_Gym_3625-1994-212.pdf
[15] http://www.statpedu.sk/Pedagogicke_dokumenty/Gymnazia/4roc/Osnovy/Informatika.doc
[16] http://www.statpedu.sk/buxus/docs//Pedagogicke_dokumenty/Gymnazia/4roc/standardy/Standard_z_informatiky.doc
[17] http://www.statpedu.sk/buxus/docs//Pedagogicke_dokumenty/Gymnazia/8roc/plan/up_G-4_3597-1990.pdf
[18] http://www.statpedu.sk/buxus/docs//Pedagogicke_dokumenty/Gymnazia/4roc/plan/informatika_gym_2311.pdf
[19] http://www.statpedu.sk/Maturita/Monitor2004/Testy/11_Test_I-1.pdf
[20] http://www.statpedu.sk/Maturita/Monitor2004/Testy/12_Test_I-2.pdf
[21] Cieľové požiadavky na vedomosti a zručnosti maturantov z informatiky, Štátny pedagogický ústav, http://www.statpedu.sk/
[22] Štátny vzdelávací program – príloha Informatická výchova ISCED1, http://www.statpedu.sk/
[23] Štátny vzdelávací program – príloha Informatika ISCED2, http://www.statpedu.sk/
[24] Štátny vzdelávací program – príloha Informatika ISCED3A, http://www.statpedu.sk/
[25] Maturitný monitor - pilotné testovanie z informatiky, http://www.statpedu.sk/
[26] Fojtík, R.: Programování na ZŠ. [online]. Dostupné na internete: http://www.ceskaskola.cz/ICTveskole/Ar.asp?ARI=1823&CAI=2129&EXPS=%22PROGRAMOV%C1N%CD%2A%22+AND+%22Z%8A%2A%22.

Tento študijný materiál vznikol ako súčasť národného projektu Ďalšie vzdelávanie učiteľov základných škôl a stredných škôl v predmete informatika v rámci Aktivity „Ďalšie vzdelávanie kvalifikovaných učiteľov informatiky na 2. stupni ZŠ a na SŠ“.

Autori © RNDr. Ľubomír Salanci, PhD.
PaedDr. Monika Tomcsányiová, PhD.
RNDr. Andrej Blaho

Názov Ďalšie vzdelávanie učiteľov základných škôl a stredných škôl v predmete informatika

Podnázov Didaktika programovania 1

Študijný materiál prešiel recenzným pokračovaním.

Recenzenti PaedDr. Daniela Bezáková, PhD.
RNDr. Gabriela Lovászová, PhD.

Počet strán 36

Náklad 400 ks

Prvé vydanie, Bratislava 2010

Vydal Štátny pedagogický ústav, Pluhová 8, 830 00 Bratislava, v súčinnosti s Univerzitou Pavla Jozefa Šafárika v Košiciach, Univerzitou Komenského v Bratislave, Univerzitou Konštantína Filozofa v Nitre, Univerzitou Mateja Bela v Banskej Bystrici a Žilinskou univerzitou v Žiline

Vytlačil BRATIA SABOVCI, s r.o., Zvolen

ISBN 978-80-8118-037-8