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

Malé programovacie jazyky

Predmet: Malé programovacie jazyky

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

Malé programovacie jazyky

Identifikácia modulu

Aktivita projektu:	1.2 Vzdelávanie nekvalifikovaných učiteľov informatiky na 2. stupni ZŠ a na SŠ
Línia aktivity:	Didaktika informatiky a informatickej výchovy
Predmet:	Malé programovacie jazyky

Garant predmetu:

RNDr. Gabriela Lovászová, PhD.
KI FPV UKF, Nitra
glovaszova@ukf.sk

Autori:

RNDr. Gabriela Lovászová, PhD., KI FPV UKF Nitra
Ing. Ľudmila Galbavá, ZŠ Benkova Nitra
PaedDr. Viera Palmárová, PhD., KI FPV UKF Nitra
PaedDr. Monika Tomcsányiová, PhD., KZVI FMFI UK Bratislava

Zaradenie modulu

Modul Malé programovacie jazyky je jediným modulom rovnomenného predmetu v línii Didaktika informatiky. Spolu s predmetom Didaktika programovania tvorí blok modulov, ktoré sa venujú metodike vyučovania programovania v rámci predmetu informatika na základných a stredných školách.

Abstrakt modulu

Malý programovací jazyk sa v tomto module chápe ako programovací jazyk, ktorý má v porovnaní s programovacími jazykmi používanými profesionálnymi programátormi menšiu množinu základných príkazov a jednoduchšiu syntax, a preto je vhodný na vyučovanie základov programovania. Malé jazyky nie sú univerzálne, sústreďujú sa na sprístupnenie nejakého programátorského princípu a tematicky sú zamerané na tvorbu programov z určitej oblasti záujmu.

Výber jazykov do tohto modulu zohľadňuje tradície vo vyučovaní programovania na Slovensku (tradičné jazyky Karel, Baltík, Logo), rôzne spôsoby zápisu programu (textový, ikonický, skladačkový), rôzne programátorské princípy (štruktúrovanosť, podprogramy, rekurzia, objekty, udalosti, paralelizmus, multimédiá), dostupnosť (softvéru a ďalších materiálov, slovenské prostredie).

Cieľom modulu je, aby sa účastníci vzdelávania zoznámili s vybranými malými programovacími jazykmi (Karel, Baltík, IzyLogo, Živý obraz, Scratch), ktoré môžu využiť na vyučovaní programovania, získali potrebné vedomosti a zručnosti, ktoré im umožnia posúdiť vhodnosť zaradenia programovania v malom programovacom jazyku do vyučovania informatiky a realizovať túto výučbu.

Obsah

Úvod

So základmi programovania sa žiaci prvýkrát stretávajú už na základnej škole na hodinách informatickej výchovy a informatiky. Našou úlohou (učiteľov) je, aby prvé kroky v programovaní boli pre žiakov sprevádzané pozitívnymi zážitkami, aby v nich prebudili záujem o túto oblasť informatiky, aby boli primerané ich veku a schopnostiam. Veľký podiel na tom, či sa nám to podarí, má výber vhodného programovacieho prostredia.

Pre „ľahký štart“ do programovania bolo vytvorených veľa programovacích prostredí, v ktorých sa na zápis programov používajú jednoduché malé programovacie jazyky. V tomto module si predstavíme päť prostredí, ktoré boli vytvorené špeciálne na vyučovanie programovania.

V module Úvod do programovania ste programovali v prostredí *Imagine Logo*

Program v jazyku Imagine Logo:

```
viem veternik
 nechfv "cervena
 opakuj 4
  [polygon
    [do 40 vp 135
     do 28 vp 45
     do 20 vp 90
     do 20]
   vp 90 ]
koniec
```

Cieľ modulu

Cieľom modulu je, aby sa účastníci vzdelávania zoznámili s vybranými malými programovacími jazykmi, ktoré môžu využiť na vyučovaní programovania. Účastníci vzdelávania majú:

- získať informácie o existencii a dostupnosti prezentovaných malých jazykov a programovacích prostredí (Karel, Baltík, IzyLogo, Živý obraz, Scratch),
- vedieť vytvárať jednoduché programy v týchto jazykoch,
- vedieť zhodnotiť, aké programátorské princípy jednotlivé malé programovacie jazyky sprístupňujú,
- byť schopní rozhodnúť o vhodnosti zaradenia niektorého malého jazyka do vyučovania a realizovať výučbu (po ďalšom samoštúdiu).

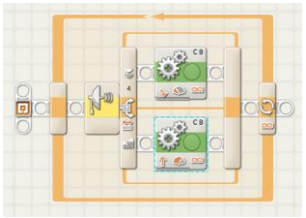
Vstupné vedomosti

Požadované prerekvizity

Účastníci vzdelávania v predchádzajúcich troch semestroch výučby absolvovali v línii Informatika predmety súvisiace s programovaním: Úvod do programovania (1 modul), Programovanie (9 modulov) a Robotické stavebnice vo vzdelávaní (1 modul). Absolvovali tiež úvodný modul predmetu Didaktika programovania v línii Didaktika informatiky.

V module Robotické stavebnice ste programovali robotov v grafickom jazyku *NXT-G*

Program v jazyku NXT-G:



Predpokladané vstupné vedomosti, skúsenosti a zručnosti

Účastníci vzdelávania sa v doterajšom štúdiu zoznámili s programovacími jazykmi Logo, Pascal a LEGO Mindstorms Education NXT. Poznajú syntax programovacích jazykov Logo a Pascal; vedia vytvárať projekty v programovacích prostrediach Imagine Logo a Delphi alebo Lazarus; vedia používať algoritmické konštrukcie štruktúrovaného programovacieho jazyka; vedia navrhovať aplikácie skladaním z komponentov riadené udalosťami; vedia vytvárať jednoduché programy na ovládanie robotov v ikonickom programovacom jazyku.

Po absolvovaní úvodného modulu predmetu Didaktika programovania vedia vymedziť postavenie programovania v rámci informatiky na základnej a strednej škole; poznajú historické súvislosti vzniku a vývoja programovacích jazykov; dokážu analyzovať a posúdiť vhodnosť programovacích jazykov a prostredí pre potreby vyučovania programovania.

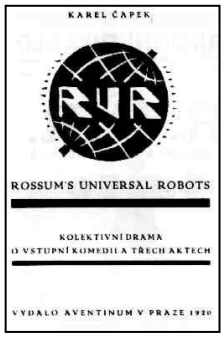
1 Karel

Programovací jazyk *Karel* vytvoril v roku 1981 **Richard Pattis** na Standfordskej univerzite v USA. Bol určený pre študentov úvodného kurzu programovania ako propedeutika štruktúrovaného programovania v jazyku Pascal.

Karel je typickým príkladom *malého programovacieho jazyka*. Obsahuje malý počet jednoduchých príkazov a nepredpokladá žiadne špeciálne predchádzajúce vedomosti či zručnosti (napr. z matematiky). Karlov mikrosvet predstavuje miestnosť, v ktorej sa robot môže pohybovať (robiť krok vpred, otáčať sa), klásť a zdvíhať tehličky (resp. kvádre a značky), zisťovať stav svojho okolia (t. j. otestovať prítomnosť steny, tehly či značky). Téma mikrosveta umožňuje sformulovať veľa zaujímavých, rôzne náročných typov úloh. Riešenie úlohy sa z pohľadu žiaka javí ako hranie hry, zábava, plnenie misie, čo významne podporuje jeho motiváciu pre učenie sa. Vizualizácia činnosti robota (pri interpretovaní príkazov programu) na obrazovke umožňuje rozvíjanie algoritmického myslenia a osvojovanie si základov programovania názorným spôsobom.

Veľkou výhodou jazyka Karel je fakt, že má syntax podobnú jazyku Pascal a prirodzene núti žiaka *programovať štruktúrovane s využitím riadiacich konštrukcií sekvencia, vetvenie a cyklus*. V editovacom režime *je možné definovať vlastné príkazy a podmienky* (bez parametrov) a pri riešení zložitejších úloh aplikovať metódu návrhu algoritmu/programu zhora nadol (resp. zdola nahor). Karel nepozná premenné ani výrazy (vzhľadom na špecifické ciele programovacieho jazyka nie sú potrebné). Pri riešení niektorých úloh sa otvára priestor na použitie *rekurzie*.

Richard E. Pattis dal svojmu programovaciemu jazyku meno **Karel** na počesť českého spisovateľa **Karla Čapka**, ktorý vo svojej hre RUR (1920) prvýkrát v histórii použil slovo *robot*.



Skutočným vynálezcom slova *robot* je však Karlov brat Josef Čapek.

http://capek.misto.cz/robot.html

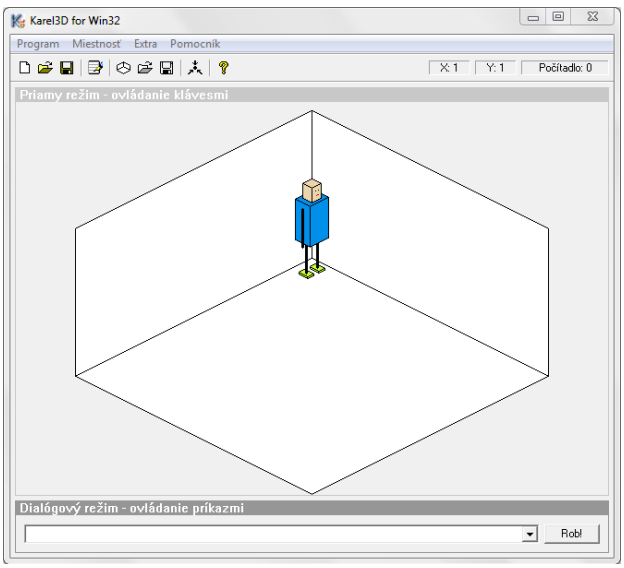
Prvé stretnutie s robotom Karlom

Po spustení programovacieho prostredia *Karel3D for Win32* vidíme na obrazovke prázdnu miestnosť základnej veľkosti s Karlom stojacim vo východiskovej pozícii. Keď klikneme na okno s miestnosťou, môžeme Karla ovládať priamo pomocou klávesov. Ak je aktívne podokno s príkazovým riadkom, pracujeme v dialógovom režime. Prostredníctvom príkazov hlavnej ponuky (resp. tlačidiel na paneli nástrojov) môžeme otvárať a ukladať miestnosti a vytvorené programy, ako aj zmeniť niektoré nastavenia prostredia.

Existujú rôzne implementácie Karla. Niektoré verzie reprezentujú Karlov mikrosvet dvojrozmernou miestnosťou (ako obdĺžnikovú sieť so štvorcovými políčkami). My použijeme atraktívnejšiu slovenskú implementáciu s trojrozmernou miestnosťou *Karel3D for Win32* [4]. Je k dispozícii v elektronickom kurze pre tento modul.

Príkazy v prostredí *Karel 3D for Win32* sa dajú zadávať v slovenčine, v angličtine a v maďarčine. Jazyk prepíname v menu *Extra/Nastavenia/Interpreter.*

Zmeňte rozmery miestnosti a farbu Karlovho trička ☺.

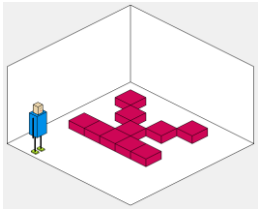


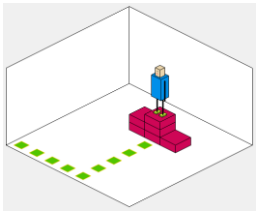
Obr. 1.1 Pohľad na programovacie prostredie Karel3D for Win32

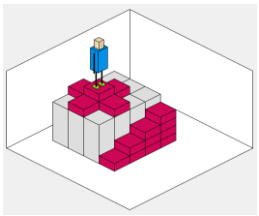
Priamy režim

V priamom režime môžeme Karla ovládať pomocou klávesov:

`šípky`	krok vpred, otočenie
`Insert`	položenie / odstránenie značky
`PgUp, PgDn`	položenie / odstránenie kvádra
`Home, End`	položenie / zdvihnutie tehličky







Úloha 1.1

V prázdnej miestnosti:

a) postavte z tehličiek písmeno K,

b) postavte z tehličiek stupeň víťazov, ku ktorému povedie chodník zo značiek,

c) z tehličiek a kvádrov postavte stavbu podľa vlastnej fantázie.

Vytvorené miestnosti uložte na disk.

Zistite, čo všetko robot Karel dokáže a čo je nad jeho sily.

Dialógový režim

V dialógovom režime píšeme príkazy (jeden príkaz alebo viac príkazov oddelených medzerami) do príkazového riadku. Karel pozná tieto príkazy:

`KROK`	urobí krok vpred
`VLAVO`	otočí sa o 90° vľavo
`VPRAVO`	otočí sa o 90° vpravo
`POLOZ`	položí pred seba jednu tehlu
`ZDVIHNI`	zdvihne tehlu, ktorú vidí pred sebou
`OZNAC`	miesto, na ktorom stojí, označí značkou
`ODZNAC`	z označeného miesta, na ktorom stojí, odstráni značku

Príkazy môžeme písať aj malými písmenami.

Interpreter príkazy napísané v príkazovom riadku číta a vykonáva postupne zľava doprava.

Úloha 1.2

V priečinku *uloha2* sú pripravené miestnosti. Postupne ich otvárajte a s využitím vyššie uvedených príkazov ich poupratujte (zozbierajte všetky tehličky).

Cyklus so známym počtom opakovaní

Karel stojí vo svojej východiskovej pozícii v rohu prázdnej miestnosti. Uhádnite, čo Karel postaví, keď mu dáme tieto príkazy:

a)

```
OPAKUJ 4 KRAT KROK *OPAKUJ
VPRAVO
OPAKUJ 4 KRAT KROK *OPAKUJ
OPAKUJ 4 KRAT KROK POLOZ VLAVO *OPAKUJ
```

b)

```
OPAKUJ 4 KRAT OPAKUJ 10 KRAT POLOZ KROK *OPAKUJ  VPRAVO *OPAKUJ
```

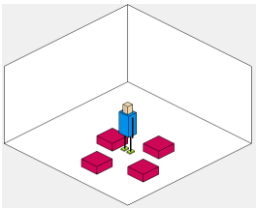
Príkaz cyklu so známym počtom opakovaní:

```
OPAKUJ počet KRAT
   príkaz1
   príkaz2
   ...
   príkazn
*OPAKUJ
```

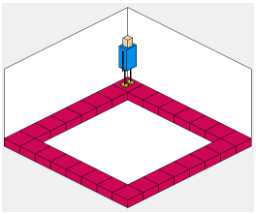
Porovnajte zápisy cyklu so známym počtom opakovaní v jazyku Karel a v jazyku Pascal.

Odpovede k hádankám zo strany 5:

a)



b)



Úloha 1.3	V priečinku *uloha3* sú pripravené miestnosti. Postupne ich otvárajte a s využitím príkazu `OPAKUJ` ich poupratujte (zozbierajte všetky tehličky).
Úloha 1.4	Navrhnite miestnosť, ktorú je výhodné poupratovať s využitím príkazu `OPAKUJ`. Nechajte ju poupratovať svojho suseda.

Definovanie vlastných príkazov

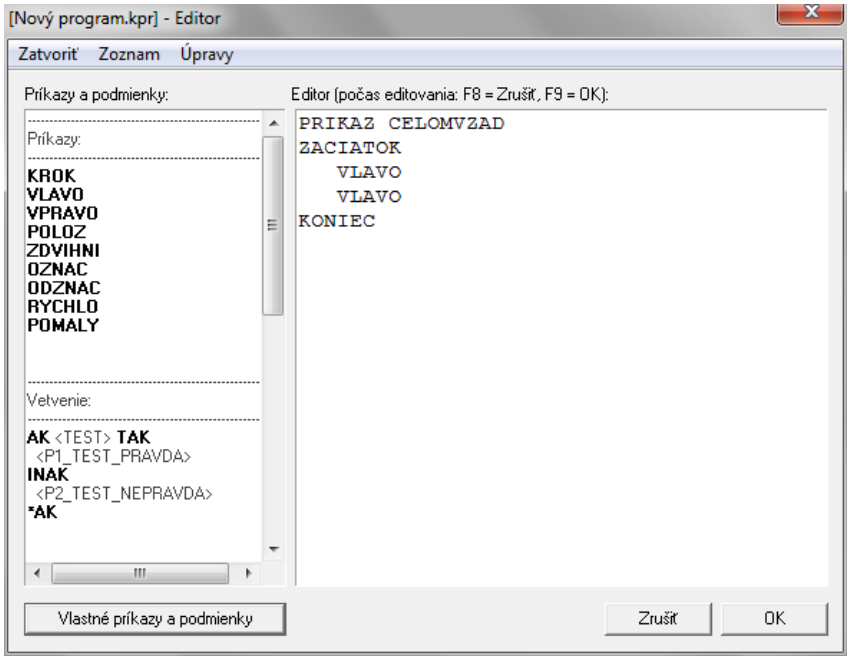
Naučíme Karla príkaz `CELOMVZAD` (otočenie o 180°).

Vlastné príkazy môžeme definovať pomocou editora, ktorý sa zobrazí po kliknutí na tlačidlo  (*Ctrl* + *E*).

V editore vidíme v ľavom okne prehľad základných príkazov a podmienok jazyka (resp. vlastných príkazov a podmienok). V pravej časti okna môžeme definovať nové príkazy:

Pri definovaní nových príkazov musíme dodržať takúto syntax:

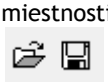
```
PRIKAZ meno_prikazu
ZACIATOK
   príkaz1
   príkaz2
   ...
   príkazn
KONIEC
```



Obr. 1.2 Okno editora na definovanie vlastných príkazov

Uloženie a otvorenie programu:

Uloženie a otvorenie miestnosti:



Po potvrdení zmien kliknutím na tlačidlo OK sa nový príkaz uloží a môžeme ho používať. Ak chceme definovať ďalší príkaz, napíšeme ho pod príkaz `CELOMVZAD`.

Vytvorený program (všetky vlastné príkazy, ktoré definujeme v editovacom režime), môžeme podobne ako miestnosť uložiť na disk. Programy prostredia *Karel3D for Win32* majú príponu *kpr*.

Úloha 1.5

Naprogramujte uvedené príkazy a otestujte ich:

a) `KROKVZAD` (robot cúva),

b) `KROKNATEHLE` (robot stojí na tehle, posunie sa v smere natočenia stojac na tehle),

c) `TLACTEHLU` (v miestnosti je jedna tehla, robot ju má tlačiť pred sebou).

V riešení úlohy 5 b) nechceme na obrazovke zbadať ako Karel

1. zostúpi z tehličky,
2. otočí sa čelom vzad,
3. zdvihne ju a urobí krok na jej miesto,
4. znovu sa otočí,
5. položí tehličku pred seba
6. a vystúpi na ňu.

Pred sekvenciou týchto príkazov preto použijeme príkaz `RYCHLO`. Vďaka nemu uvidíme až výsledný stav po vykonaní všetkých príkazov. Bude sa nám zdať, že Karel sa posunul stojac na tehličke.

K pôvodnému nastaveniu interpretera sa vrátime príkazom `POMALY`.

Úloha 1.6

Vpravo vidíte dve miestnosti s rovnakým počtom tehličiek. Napíšte čo najkratšie programy na ich upratanie.

Úloha 1.7

Sformátujte uvedený príkaz s vhodným logickým odsadzovaním:

```
OPAKUJ 2 KRAT OPAKUJ 2 KRAT KROK *OPAKUJ
VPRAVO OPAKUJ 3 KRAT POLOZ KROK *OPAKUJ VLAVO
*OPAKUJ
```

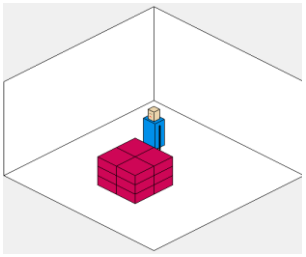
Úloha 1.8

Preštudujte uvedený príkaz. Odhadnite výsledok, ktorý na obrazovke uvidíte po jeho vykonaní. Predpokladajte, že Karel stojí uprostred prázdnej miestnosti.

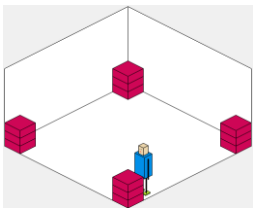
```
PRIKAZ KUZLO
ZACIATOK
  RYCHLO
  OPAKUJ 10 KRAT
     POLOZ
     KROK
     CELOMVZAD
     POLOZ
     KROK
     CELOMVZAD
  *OPAKUJ
  OPAKUJ 10 KRAT
     ZDVIHNI
  *OPAKUJ
  POMALY
KONIEC
```

Miestnosti k *úlohe 1.6*:

a)



b)

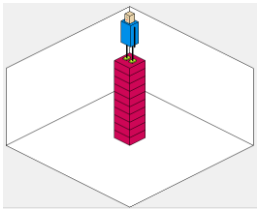


Zamyslite sa:

„Prečo je dôležité ako je program sformátovaný? Veď počítaču na tom nezáleží...“

Odpoveď na hádanku z úlohy 1.8:

Karel sa po vykonaní príkazu `KUZLO` ocitne na vrchu stĺpika z desiatich tehličiek.



Úloha 1.9

Definujte uvedené príkazy a otestujte ich:

a) `STLP` (robot pred sebou postaví stĺpik z tehličiek),

b) `ZAMURUJSA` (robot sa obstavia stĺpmi tak, že si zablokuje cestu von),

c) `BAZEN` (robot urobí po okrajoch miestnosti múriky),

d) `KOBEREC` (robot vydláždi celú miestnosť tehličkami).

Podmienky, cyklus s podmienkou, vetvenie

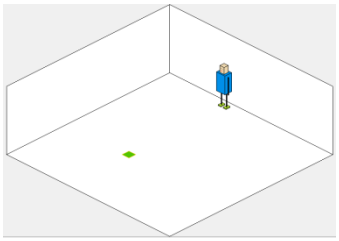
Vyriešme spoločne tieto Karlove problémy:

a)

Karel stojí pri jednej stene miestnosti a niekde pred ním je značka. Napíšeme príkaz `CHODNAZNACKU`, po vykonaní ktorého sa Karel ocitne na označenom mieste:

Zapíšte cyklus s podmienkou na začiatku v jazyku *Pascal*.

```
PRIKAZ CHODNAZNACKU
ZACIATOK
  KYM NIEJE ZNACKA ROB
     KROK
  *KYM
KONIEC
```



b)

Karel stojí na chodníku z tehličiek. Na niektorých miestach je chodník deravý (chýba v ňom tehlička). Napíšeme príkaz `OPRAVCHODNIK` tak, aby po jeho vykonaní stál Karel na opačnom konci chodníka a všetky medzery v chodníku boli opravené.

Zoznam podmienok:

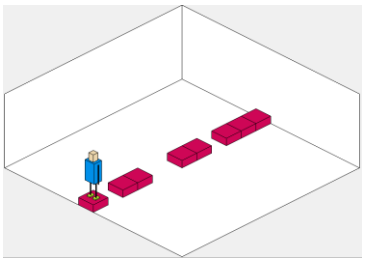
```
ZNACKA
TEHLA
VOLNO
STENA
```

Platnosť podmienok testujeme takto:

```
JE ZNACKA
JE TEHLA
JE VOLNO
JE STENA

NIEJE ZNACKA
NIEJE TEHLA
NIEJE VOLNO
NIEJE STENA
```

```
PRIKAZ OPRAVCHODNIK
ZACIATOK
  KYM NIEJE STENA ROB
    AK JE TEHLA TAK
       KROK
    INAK
       POLOZ
       KROK
    *AK
  *KYM
KONIEC
```



Upravíme príkaz `OPRAVCHODNIK` tak, aby neobsahoval vetvu `INAK`.

```
PRIKAZ OPRAVCHODNIK
ZACIATOK
  KYM NIEJE STENA ROB
    AK NIEJE TEHLA TAK
      POLOZ
    *AK
    KROK
  *KYM
KONIEC
```

Úloha 1.10

V priečinku *uloha10* nájdete miestnosti, v ktorých má Karel

a) dotlačiť tehličky na miesta so značkami,

b) preložiť všetky stĺpiky z tehličiek za seba (stĺpiky nie sú rovnako vysoké).

Navrhnite a naprogramujte vlastné príkazy, pomocou ktorých vyriešite Karlove problémy.

Úloha 1.11

V priečinku *uloha11* nájdete miestnosti, v ktorých má Karel

a) vedľa každého stĺpika z tehličiek postaviť rovnako vysoký stĺpik,

b) vydláždiť chodníky od svojej východiskovej pozície (označenej značkou) ku všetkým stenám miestnosti,

c) opraviť múr okolo svojho pozemku doplnením chýbajúcich tehličiek,

d) postaviť schody vedúce na vrch kvádra.

Navrhnite a naprogramujte vlastné príkazy, pomocou ktorých vyriešite Karlove problémy.

Pri riešení náročnejších úloh je výhodné definovať si vlastné zložené podmienky, napr.:

```
PODMIENKA PREKAZKA
ZACIATOK
  AK JE STENA TAK
    PRAVDA
  INAK
    AK JE TEHLA TAK
      PRAVDA
    INAK
      NEPRAVDA
    *AK
  *AK
KONIEC
```

V ďalších príkazoch potom môžeme testovať, či je podmienka `PREKAZKA` splnená alebo nie:

```
JE PREKAZKA
```

resp.

```
NIEJE PREKAZKA
```

Rekurzia

Otestujme tento príkaz:

```
PRIKAZ TOCSANAMIESTE
ZACIATOK
   VPRAVO
   TOCSANAMIESTE
KONIEC
```

Karel po spustení tohto príkazu stojí na mieste a opakovane („donekonečna“) sa otáča vpravo. Zastavíme ho kliknutím na tlačidlo STOP! pri príkazovom riadku.

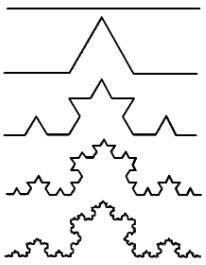
Príkaz, ktorý vo svojom tele obsahuje volanie seba samého, nazývame *rekurzívny*. Ak je rekurzívne volanie uvedené ako posledný príkaz v poradí, hovoríme o tzv. *chvostovej rekurzii*.

Otestujme teraz tento príkaz:

```
PRIKAZ BEHAJDOKOLA
ZACIATOK
   AK JE STENA TAK
     VPRAVO
   *AK
   KROK
   BEHAJDOKOLA
KONIEC
```

Čo je to *rekurzia*?

Objekt je rekurzívny, ak sa čiastočne skladá alebo je definovaný pomocou seba samého.



Úloha 1.12

Upravte príkaz `TOCSANAMIESTE` tak, aby sa Karel otáčal opačným smerom a pri otáčaní si so sebou prenášal tehličku (stále ju chce vidieť pred sebou).

V matematike sa stretávame s rekurzívnymi definíciami, napr.:

```
0! = 1
n! = n.(n-1)!
```

V programovaní nazývame rekurzívnym taký podprogram, ktorý vo svojom tele volá sám seba.

Vo vetve `else` funkcie na výpočet hodnoty faktoriálu voláme tú istú funkciu `fakt`, ako parameter ale uvádzame hodnotu `n-1`:

```
function fakt(n:integer):integer;
begin
  if n=0 then result:= 1
  else result:= n * fakt(n-1);
end;
```

Ako bude prebiehať výpočet pre volanie `fakt(4)` ?

Koľkokrát sa pri ňom zavolá funkcia `fakt`?

```
4! = 4 . 3!
     3 . 2!
       2 . 1!
         1. 0!
            1
```

Úloha 1.13	Upravte príkaz `BEHAJDOKOLA` tak, aby Karel okolo celej miestnosti nielen behal, ale zároveň popri stenách budoval múr z tehličiek.
Úloha 1.14*	Chvostová rekurzia sa dá prepísať na cyklus (prečo?). Upravte príkazy z úloh 1.12 a 1.13 tak, aby neboli rekurzívne.

Vytvorme príkaz, pri vykonávaní ktorého robot prejde ku stene a vráti sa späť na pôvodné miesto bez toho, aby si ho predtým označil.

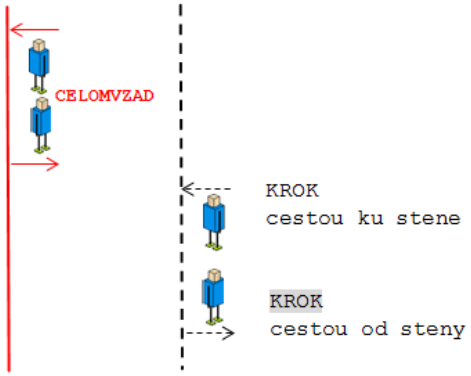
Riešenie:

Ako by ste tento problém riešili v jazyku Pascal? Pravdepodobne by ste si v pomocnej premennej pamätali počet krokov, ktoré bolo treba urobiť ku stene a o rovnaký počet krokov by ste Karla poslali späť. *Karel ale nepozná premenné* a preto problém musíme vyriešiť iným spôsobom. Urobíme to s využitím rekurzie:

```
PRIKAZ KUSTENEASPAT
ZACIATOK
    AK JE STENA TAK
       CELOMVZAD
    INAK

       KROK
       KUSTENEASPAT
       KROK

   *AK
KONIEC
```



Podfarbený príkaz `KROK` bude Karel prvýkrát vykonávať až potom, čo sa pri stene otočí čelom vzad. Príkaz `KROK` sa celkovo vykoná práve toľkokrát, koľkokrát sa predtým zavolal rekurzívne príkaz *KUSTENEASPAT*. Každé volanie tohto príkazu sa totiž musí vykonať až do konca. Vždy, keď skončí jedno rekurzívne volanie *KUSTENEASPAT*, vo vetve `INAK` predošlého volania sa pokračuje príkazom `KROK`.

Úloha 1.15*	Karel stojí pred múrom z tehál ľubovoľnej výšky. Naprogramujte príkaz `CEZMUR`, ktorým Karel prejde cez múr tak, že si vybúra otvor, prejde a otvor za sebou zamuruje do pôvodnej výšky.

Čo sme sa naučili

Spoznali sme príkazy jazyka Karel a vieme robota riadiť v priamom aj dialógovom režime. V editovacom režime sme vytvorili viacero vlastných príkazov, ktoré sme využili pri riešení konkrétnych problémov. Naučili sme sa používať cykly aj vetvenie. Pri písaní príkazov sme dodržiavali logické odsadzovanie. Sme schopní navrhnúť vlastné úlohy na upratovanie miestností. Zoznámili sme sa s možnosťou využiť pri riešení problémov programovaciu techniku rekurzia. Vieme porovnať jazyky Karel a Pascal.

2 Baltík

SGP Baltík 3 je detský štruktúrovo orientovaný programovací jazyk. Je to produkt českej softvérovej firmy SGP Systems (Soukup Graphics Programming Systems), jeho autorom je Bohumír Soukup. Tento jazyk disponuje príkazmi na úrovni jazyka typu C, ktoré sú zobrazené v ikonografickej podobe, čo zaručuje tomuto nástroju ľahkú ovládateľnosť a použiteľnosť už vo veľmi nízkom veku žiakov. Má viacero funkcií, je to multimediálny programátorský a kresliaci nástroj. Hlavnou postavou je čarodejník Baltík, ktorý dokáže vykonávať rôzne príkazy a čarovať predmety.

Na zoznámenie s Baltíkom použijeme slovenskú demoverziu SGP Baltík 3.7 SK, ktorá je voľne prístupná na internete [6]. Demoverzia neumožňuje ukladať súbory. Vyskúšame si prostredie, prácu v jednotlivých režimoch a niektoré druhy príkazov.

Zoznámte sa, Baltík:

Úvod

Baltík má tri základné úrovne - režimy, z toho prvé dva (**Skladať scénu** a **Čarovať scénu**) sú prípravné režimy na programovanie. Tretí režim **Programovať** má dve úrovne (začiatočník a pokročilý). Medzi režimami sa prepíname kliknutím na záložku s názvom režimu (v menu *Režimy*).

SGP Baltík 3 je platený softvér. Väčšina slovenských základných škôl však dostala v minulosti od Infoveku licenciu na tento program zadarmo spolu s učebnicou [5]. Je možnosť kúpiť rôzne druhy ďalších licencií, napr. ročné licencie pre žiakov na doma. Bližšie informácie nájdete na stránke autorskej firmy SGP [6] v časti „Vše o licencích“.

Úloha 2.1	Vyskúšajte si programy *Hovoriace počty* a *Katarína*. Sú to multimediálne programy vytvorené v Baltíkovi a sú súčasťou distribúcie Baltíka (*Štart - Všetky programy - SGP Systems - Výber*).
Úloha 2.2	Spustite a pozrite si program *Audiovizuálny sprievodca Baltíkom 3* (*Štart - Všetky programy - SGP Systems* alebo tento program nájdete aj v samotnom Baltíkovi v menu *Pomoc*).

Okrem SGP Baltíka 3 existuje aj programovací nástroj **SGP Baltie 4 C#**. Je to moderné objektovo orientované programovacie prostredie založené na DirectX a .NET. Obsahuje množstvo režimov vhodných pre úplných začiatočníkov (interaktívne režimy vhodné už od 4 rokov) až po pokročilých programátorov (režim C# bez ikoniek pre písanie ľubovoľného programu v C#). Umožňuje jednoduchým spôsobom atraktívne programovanie v 3D prostredí. Je to tiež platený softvér, demoverziu je možné stiahnuť na stránke SGP [6].

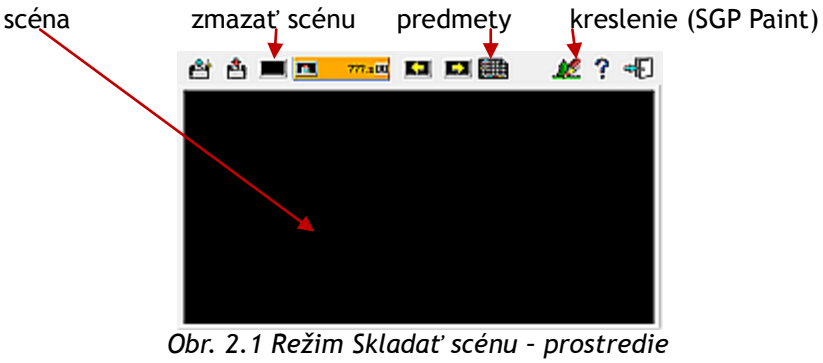
Takto vyzerá Baltík v 3D prostredí SGP Baltie 4 C#:

Režimy Skladať scénu a Čarovať scénu

Režim Skladať scénu

V menu klikneme na *Režimy* a vyberieme *1. Skladať scénu*. Tento najjednoduchší režim zvládajú už deti od 3 – 4 rokov. Umožňuje skladať obrázky pomocou myši a kresliť nové predmety. Neskôr takto vytvorené scény slúžia ako grafické podklady pre programy. Pre prácu v tomto režime potrebujeme poznať nasledovné pojmy a činnosti:

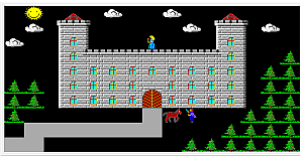
- **scéna** – pracovná plocha, na ktorú pomocou myši môžeme umiestniť predmety (maximálne 150, rozmiestnených do 10 riadkov a 15 stĺpcov). Uložená scéna je súbor s príponou *Sxx*, kde *xx* je číslo scény, napr. hrad.S00, hrad.S01, dom.S00.



Obr. 2.1 Režim Skladať scénu - prostredie

V plnej verzii programu SGP Baltík 3 si pomocou nástroja SGP Paint môžeme nakresliť nové vlastné banky predmetov.

Poskladaná scéna:

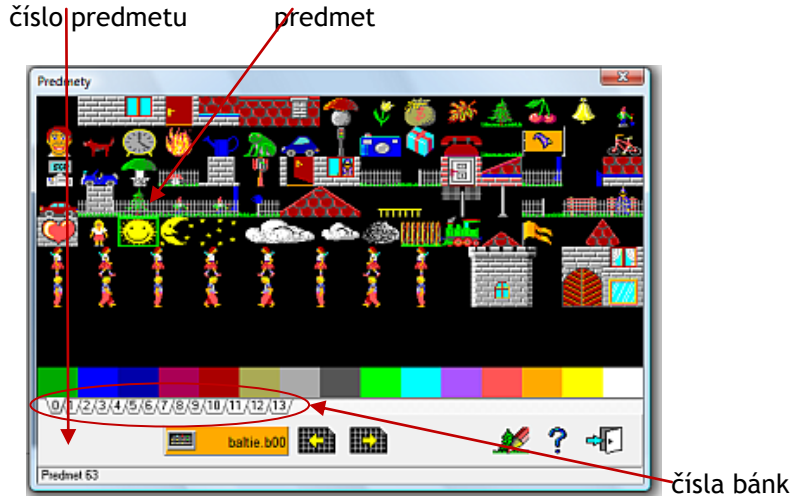


SGP Baltík 3 je nástroj, ktorý sa často používa na rozvíjanie algoritmického myslenia a výučbu programovania na základných školách. Je vhodný na použitie už od 1. triedy základných škôl. Je rozšírený na školách hlavne v Česku, Slovensku a Poľsku. Postupne sa pridávajú ďalšie štáty. Každoročne sa organizuje niekoľko medzinárodných súťaží, do ktorých sa zapájajú žiaci už od útleho veku. Je možnosť zapojiť sa napr. do súťaže:

- Creative Baltie - tradičná medzinárodná autorská tvorivá súťaž,
- Baltie – súťaž, v ktorej sa riešia zadané úlohy, má niekoľko kôl – od školského až po medzinárodné,
- Baltie Junior (predtým Baltíkov Borec) - medzinárodná súťaž pre začínajúcich programátorov,
- Christmas Baltie (Vianoce s Baltíkom) - medzinárodná autorská tvorivá súťaž.

Súťaží sa v niekoľkých vekových kategóriách, ktoré pokrývajú všetky ročníky základnej a strednej školy. Súťaží sa v jazykoch SGP Baltík 3 aj SGP Baltie 4 C#. Organizátorom je autorská firma SGP, na Slovensku v spolupráci s občianskym združením TIB [7]. Firma SGP prevádzkuje špeciálny súťažný portál – bližšie na [6] v časti “Soutěže”.

- **predmet** – obrázok (má veľkosť 39 x 29 bodov). Predmety sú umiestnené v bankách predmetov, každý predmet má svoje číslo.
- **banky predmetov** – obsahujú predmety. Každá banka môže obsahovať až 150 predmetov. Prostredie Baltíka obsahuje 14 bánk, sú to súbory s názvami *baltie.b00* až *baltie.b13*.



Obr. 2.2 Banka predmetov

- **vloženie predmetu do scény** – klikneme na tlačidlo *Predmety*  , kliknutím na záložku vyberieme požadované číslo banky a klikneme na vybraný predmet, čím sa vrátime do scény a predmet umiestnime na požadované miesto.
- **presunutie predmetu** – pomocou ľavého tlačidla myši
- **kopírovanie predmetu** – pomocou pravého tlačidla myši
- **zmazanie predmetu** – pomocou ľavého tlačidla myši uchopíme predmet, presunieme mimo scény a pustíme tlačidlo myši

Úloha 2.3	Poskladajte scénu s využitím konkrétnych predmetov z bánk predmetov (napr. hrad, elektrický obvod, chemický vzorec a pod.).

Režim Čarovať scénu

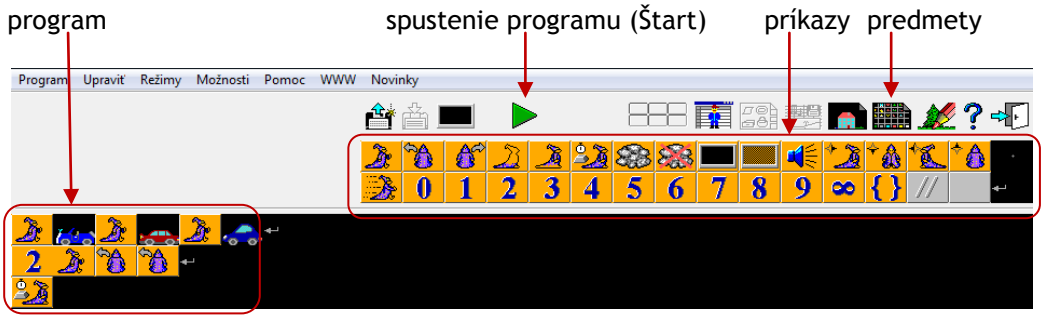
V menu klikneme na *Režimy* a vyberieme *2. Čarovať scénu*. Na scéne sa objaví čarodejník Baltík, ktorého ovládame pomocou príkazov **posuň**  , **vľavo v bok**  , **vpravo v bok**  a **vyber a čaruj**  . Baltík čaruje vybrané predmety vždy pred seba – tam, kam je otočený. Zvládnutie pohybu Baltíka a čarovania predmetov v tomto interaktívnom režime je prípravou na programovací režim.

Úloha 2.4	Vyskúšajte si pohyb Baltíka po scéne. Urobte si preteky: čí Baltík obehne prvý dookola celú scénu?
Úloha 2.5	Nech Baltík postaví domček, ktorý bude mať dvere, aspoň päť okien, strechu a komín.

Režim Programovať

Režim Programovať – začiatočník

V menu klikneme na *Režimy* a vyberieme *3. Programovať - Začiatočník*. Je to režim, v ktorom je obmedzené množstvo príkazov zameraných predovšetkým na ovládanie Baltíka.



Obr. 2.3 Režim Programovať začiatočník - prostredie

Program sa skladá z postupnosti príkazov – ikon, ktoré presúvame pomocou myši do čiernej pracovnej plochy a ukladáme ich do riadkov. Snažíme sa, aby program bol prehľadný, t.j. aby riadky neboli zbytočne dlhé a podľa možnosti tvorili menšie logické celky. Pomocou ľavého tlačidla myši príkazy presúvame, pomocou pravého tlačidla kopírujeme. Príkaz zmažeme presunutím mimo pracovnú plochu. Program spustíme tlačidlom *Štart*. Príkazy sa vykonajú postupne – zľava doprava, zhora nadol.

Zoznam niektorých príkazov:

posun Baltíka o jedno pole vpred

otočenie Baltíka o 90° vľavo, resp. vpravo

nastavenie viditeľnosti Baltíka: bude odteraz neviditeľný, resp. viditeľný

čakanie na stlačenie klávesu alebo tlačidla myši. Ak je za prvkom číslo, napr. 3 0 0 0, program bude čakať max. 3 sekundy (číslo je v milisekundách)

čarovanie predmetov s alebo bez sprievodného obláčika (štandardne je čarovanie s obláčikom)

zmazanie obrazovky

nastavenie rýchlosti, môže byť 0 – 9 a nekonečno. Napr. 7. Štandardne je rýchlosť nastavená na 5

koniec riadka, slúži na členenie programu do viacerých riadkov

blok príkazov, ohraničuje skupinu príkazov, ku ktorej sa bude Baltík správať ako k jedinému príkazu

Na internete je dostupné množstvo študijného a metodického materiálu k jazyku SGP Baltík 3 aj k prostrediu SGP Baltie 4 C#. Odkazy na tieto materiály nájdete na stránkach SGP [6], TIB - slovenské občianske združenie [7], TIB - české občianske združenie [8]. Okrem toho existuje učebnica (len v papierovej podobe) [5].

Uložený program je súbor s príponou *.bpr*, napr. *hra.bpr*. Súčasťou projektu môže byť množstvo ďalších súborov, všetky sa musia volať hra, líšia sa príponami: scény (majú príponu *Sxx*, xx-číslo scény), vlastné banky predmetov (majú prípony *Bxx*, xx-číslo banky). Celý projekt je možné uložiť aj ako tzv. **.bzip* (Baltíkov zazipovaný projekt, obsahuje všetky súbory patriace k projektu). Dostupné len v plnej verzii.

Pomoc vyvoláme kliknutím na tlačidlo *Pomoc* alebo presunutím niektorého prvku (príkazu) na toto tlačidlo.

Shift + ľavé tlačidlo myši umožňuje označenie bloku príkazov.

Na písanie komentárov - poznámok k činnosti programu - slúži prvok **komentár**. Po vložení do programu sa dá doňho písať.

Prvok **riadkový komentár** - príkazy od miesta vloženia tohto prvku až do konca logického riadku sa nevykonajú.

Niektoré ďalšie príkazy:


otočenie Baltíka na západ, juh, východ, sever


priehľadnosť (pri zobrazovaní predmetov, scén a bánk)

Úloha 2.6

Vytvorte program podľa obrázka 2.3. Skúste k nemu pridať niekoľko ďalších príkazov, napr. vyčarovanie ďalších predmetov, presun Baltíka na iné miesto scény a pod. Vyskúšajte, ako sa bude program správať, keď na jeho koniec nedáte príkaz  .

Počítané opakovanie a blok príkazov

Ak chceme, aby sa Baltík posunul napr. o 3 políčka vpred, môžeme to urobiť takto:  , alebo použijeme tzv. **počítané opakovanie:**  .

Úloha 2.7

Otvorte program *cestička.bpr* (je pripravený v e-learningovom kurze). Je v ňom načítaná scéna – domček a k nemu vedúca cestička. Dokončite program tak, aby sa Baltík dostal po cestičke ku dverám domčeka, otvoril ich (čiže vyčaroval predmet znázorňujúci otvorené dvere), vošiel dnu, dvere zatvoril a objavil sa v niektorom okne. Tip: Keď Baltík vojde do dverí, zneviditeľníme ho, objavenie v okne docielime tak, že namiesto niektorého okna vyčarujeme predmet č. 38 (Baltík v okne).

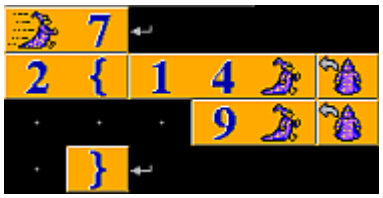
Rovnaké zadanie si vyskúšajte na programe *cestička2.bpr*, v ktorom je pripravená iná scéna.

Hotové riešenia si môžete porovnať s riešeniami v e-learningovej časti kurzu (*cestička_riešenie.bpr, cestička2_riešenie.bpr*).

Ak chceme, aby Baltík vyčaroval napr. tri počítače, môžeme to urobiť takto:  , alebo použijeme **počítané opakovanie** v spojení s **blokom príkazov**  . Ide vlastne o jednoduchý druh cyklu.

Vytvorme program, v ktorom Baltík rýchlosťou 7 obehne dookola celú scénu. Keďže scéna má rozmery 15 x 10 políčok, Baltík musí prebehnúť 14 políčok, otočiť sa vľavo, prebehnúť 9 políčok, otočiť sa vľavo a to celé ešte raz. S použitím bloku príkazov bude program vyzerať takto:

Na odsadzovanie príkazov v riadkoch používame prvok **medzera**  .



Chceme, aby pri ďalšom obehnutí obrazovky Baltík vyčaroval dookola scény stromy. Tu už použijeme vnorené bloky príkazov:

Režim programovať – pokročilý

V menu klikneme na *Režimy* a vyberieme *3. Programovať - Pokročilý*. Je to režim, v ktorom je okrem príkazov špecifických pre Baltíka aj sada príkazov bežného programovacieho jazyka typu C. Prácu v tomto režime si vyskúšame na niekoľkých riešených príkladoch.

Pomocník

Rozšírme program, v ktorom sme vyčarovali stromčeky dookola scény. Do stredu scény vyčarujme ešte tri rovnaké domčeky. Keďže domček sa skladá z viacerých predmetov a nedá sa vyčarovať naraz jedným príkazom, môžeme na vyčarovanie jedného domčeka použiť **pomocníka** (čiže procedúru alebo podprogram). Klikneme na prvok pomocníci  , ktorý sa nachádza v hornej časti obrazovky. Otvorí sa tzv. Tabuľka pomocníkov, pomocou ktorej sa pomocníci definujú a vkladajú do programu.

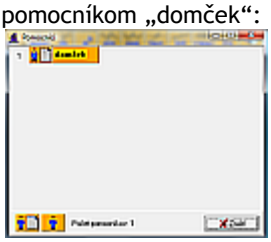
Kliknutím na prvok **Nový pomocník**  definujeme nového pomocníka, ktorého nazveme napr. „domček“. Pomocníci sa vkladajú vždy na koniec programu (za hlavný program, z ktorého sú volaní). Názov pomocníka napíšeme pomocou prvku **literál**  (tento prvok vložíme za ikonu „Nový pomocník“ a napíšeme názov). Tým je vytvorená hlavička pomocníka. Za hlavičku vložíme príkazy, ktoré budú tvoriť telo pomocníka, čiže príkazy, pomocou ktorých Baltík postaví jeden domček.

V hlavnom programe vložíme na príslušné miesto prvok **Vložiť volanie pomocníka**  (je v tabuľke pomocníkov) s označením pomocníka 

. Keďže domčeky potrebujeme tri, zavoláme ho trikrát 

.

Tabuľka pomocníkov s definovaným jedným pomocníkom „domček“:



Pomocníka môžeme označiť tromi spôsobmi:

1. textom 

2. číslom 

3. predmetom 

Úloha 2.8

Dokončite program - nech Baltík do stredu scény vyčaruje tri rovnaké malé domčeky. Môžu to byť jednoduché domčeky z dvoch predmetov ako v našom pomocníkovi alebo zložitejšie ako tie na obrázku vedľa. Na vyčarovanie jedného domčeka použite pomocníka. Riešenie si môžete porovnať s programom *pomocnik.bpr*, ktorý je pripravený v e-learningovom kurze.

Projekt Náhodná krajinka

Chceme vytvoriť program, ktorý vytvorí krajinu poskladanú z náhodného počtu predmetov rôzneho druhu (stromy, hríby, oblaky a pod.). Na to budeme potrebovať príkazy súradníc, príkaz náhodné číslo a literál.

Baltík má dva druhy **súradníc**:

- **políčkové**  - súradnice políčok označujeme veľkými písmenami X, Y, kde X označuje číslo stĺpca a Y číslo riadku. Predmety môžeme umiestniť priamo na konkrétne súradnice. Napr. príkaz 

vyčaruje strom do 2. stĺpca a 9. riadku scény.
- **bodové**  - súradnice bodov označujeme malými písmenami x, y. Napr. príkaz 

vyčaruje strom na konkrétny bod (na bode 250, 100 bude umiestnený ľavý horný roh predmetu).

Aby príkaz nebol zbytočne dlhý, na zadanie súradníc x, y sme použili prvok **literál**  . Používa sa na priame zadanie čísla (môžeme ho použiť všade tam, kde potrebujeme čísla) alebo reťazca (už sme ho použili napr. na zadanie názvu

Baltíkova scéna má veľkosť 15 x 10 políčok. Má teda 15 stĺpcov, ktoré číslujeme od 0 po 14 (X) a 10 riadkov, ktoré číslujeme od 0 po 9 (Y).

Scéna má veľkosť 585 x 290 bodov (pretože 1 políčko má veľkosť 39 x 29 bodov). Má teda 585 stĺpcov, ktoré číslujeme od 0 po 584 (x) a 290 riadkov, ktoré číslujeme od 0 po 289 (y).

Po vložení ikony **literál**  do programu sa objaví kurzor a dá sa do nej písať. Ak zadáme celé číslo, bude mať azúrovú farbu, ak reálne, bude zelená, ak text, bude žltá. Ak chceme zmeniť už vytvorený literál, ukážeme naň kurzorom myši a stlačíme F2 alebo klikneme na literál pravým tlačidlom myši a zvolíme „Upraviť“. Takto je možné, kliknutím na štvorček s príslušnou farbou, zmeniť aj typ literálu, ak je to možné vzhľadom na vložený obsah (napr. z čísla na reťazec alebo naopak).

pomocníka).

Potrebujeme ešte príkaz **náhodné číslo**  . Napr. príkaz  vráti náhodné číslo z intervalu 0 – 14, príkaz  vráti náhodné číslo z intervalu 3 – 5. Príkaz  vyčaruje strom náhodne na niektoré políčko scény.

Teraz vytvorme program, v ktorom využijeme popísané príkazy. V našej krajinke bude les s náhodným počtom stromov (30 – 50), náhodným počtom hríbov (10 – 15), náhodným počtom líšok (1 – 3) a jednou sovou. Nad lesom bude obloha s oblakmi (10 – 20), vtákmi (3 – 5) a slnkom. Všetky predmety budú umiestnené na náhodné súradnice. Zabezpečíme však, aby les a obloha boli oddelené (zvolíme vhodný interval čísiel riadkov, čiže súradnice Y). Použijeme políčkové súradnice. Začiatok programu môže vyzerať napr. takto:

Riešenie nájdete aj v e-learningovom kurze pod názvom *náhodná krajinka.bpr*.

Úloha 2.9

Dokončite program podľa vyššie uvedeného zadania tak, aby výsledná krajinka bola podobná tej na obrázku vľavo. Potom skúste meniť zobrazované predmety, ich počet, prípadne umiestnenie.

Projekt Bludisko

V tomto projekte ukážeme, ako naprogramovať jednoduchú hru. Chceme, aby sme pohyb Baltíka ovládali kurzorovými šípkami na klávesnici. Cieľom hry je, aby Baltík pozbieral všetky balíčky, ktoré sú rozmiestnené v bludisku. Pripravili sme štyri spôsoby, od najjednoduchšieho k zložitejšiemu, ako takýto program vytvoriť:

Bludisko 1

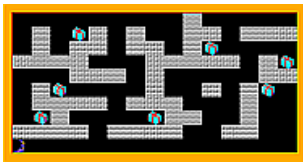
V programe je načítaná scéna s bludiskom, pohyb Baltíka je ovládaný kurzorovými šípkami, klávesom Space sa „berú“ balíčky. V programe nie sú žiadne kontroly, Baltík sa môže voľne pohybovať po ploche a „brať“ čokoľvek.

Ovládanie pohybu Baltíka môžeme naprogramovať ako nekonečný cyklus, v ktorom sa program rozvetvuje podľa stlačeného klávesu. V Baltíkovi sa viacnásobné rozvetvenie robí zdvojením prvku **if** (keď)  . Pomocou prvku **nejaký kláves**  budeme testovať, aký kláves bol stlačený. Program môžeme začať tvoriť napr. takto:

Pozrite si program *náhodná krajinka2.bpr*. V programe sme zmenili typ súradníc na bodové a pretože predmety sa prekrývajú, nastavili sme priehľadnosť čiernej farby (pomocou prvkov **priehľadnosť**  a **farba**).

Ak zmeníme políčkové súradnice na bodové, výsledok vyzerá takto:

Bludisko:



Ak stlačíme kláves *šípka hore*, Baltík sa otočí na sever a posunie o 1 políčko vpred. Ak stlačíme kláves *Space*, Baltík vyčaruje predmet č. 0. Predmet č. 0 je špeciálny prípad – tam, kde sa vyčaruje, bude zrušený pôvodný predmet (ak tam nejaký je) a nebude tam nič, je to tzv. prázdne políčko.

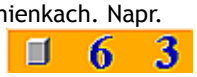
Ikony klávesov nájdete, ak kliknete na ikonu *Klávesy, konštanty, premenné* v hornej časti obrazovky (prvá záložka okna, ktoré sa otvorí).

Úloha 2.10

Otvorte program *bludisko.bpr*, ktorý je v e-learningovom kurze. Je v ňom pripravená scéna – bludisko. Dokončite program podľa vyššie uvedeného zadania tak, aby bol Baltík ovládaný všetkými štyrmi kurzorovými šípkami, t.j. aby sa mohol pohybovať do všetkých svetových strán a aby klávesom *Space* zbieral balíčky.

Bludisko 2

V tejto verzii už Baltík nemôže chodiť cez steny bludiska ani cez balíčky (čiže nemôže vstúpiť na žiadny predmet, môže chodiť iba po chodníkoch v bludisku). Môže „zobrať“ iba predmet balíček.

Príkaz **nejaký predmet**  umožňuje okrem iného prevod čísla na predmet, ktorý potom môžeme vyčarovať alebo otestovať jeho výskyt v podmienkach. Napr. príkaz  vyčaruje predmet  . Číslo predmetu zistíme, ak naň v banke predmetov nastavíme kurzor myši. Jeho číslo sa zobrazí v ľavom dolnom rohu okna.

Úloha 2.11

Upravte program *bludisko.bpr* tak, aby spĺňal požiadavky zadania v *Bludisku 2*. Na to bude potrebné:

- vždy, keď sa má Baltík pohnúť o políčko vpred, otestovať, či nie je pred ním nejaký predmet. Použije sa napr. príkaz **if**  . Ak chcete, môžete si na tento účel vytvoriť pomocníka.
- vždy, keď sa stlačí kláves *Space*, testovať, či je pred Baltíkom predmet balíček:  .

Bludisko 3

V hre sa už sleduje počet pozbieraných balíčkov (pomocou celočíselnej globálnej premennej, jej obsah sa vypisuje do pravého horného rohu scény). Keď sú pozbierané všetky, zobrazí sa nápis „KONIEC“ a po stlačení klávesu alebo tlačidla myši program skončí. Hru sme zrýchlili nastavením rýchlosti na 7.

Budeme potrebovať vedieť:

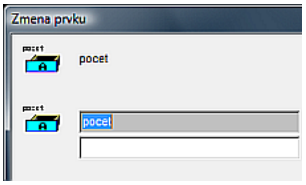
- použiť celočíselnú globálnu premennú,
- vypísať na obrazovku text a obsah premennej,
- predčasne ukončiť cyklus (použijeme príkaz **break** ).

Podľa typu delíme premenné Baltíka na tri druhy:
- celočíselné (azúrové),
- reálne (zelené),
- reťazcové (žlté).

Podľa rozsahu platnosti (viditeľnosti) pozná Baltík premenné:
- globálne (zásuvky),
- lokálne (košíky).

Premenné v Baltíkovi majú tvar ikoniek, sú uložené v špeciálnych bankách premenných. Nájdeme ich pod ikonou *Klávesy, konštanty, premenné*  . V okne, ktoré sa otvorí, zvolíme záložku *Celočíselné globálne premenné*  . Ikony majú tvar zásuviek. Kliknutím vyberieme niektorú z nich, napr. A. V okne *Zmena prvku* máme možnosť nastaviť rôzne parametre premennej, napr. meno, počiatočnú hodnotu, atď. Ak počiatočnú hodnotu nezadáme, je počiatočná hodnota automaticky nula.

Celočíselné globálne premenné:



Výpis textu na obrazovku – text (čiže reťazcový literál) môžeme priradiť na

Ukážky operácií s premennou:

 - zvýšenie hodnoty premennej o 1 (ak zadáme za šípku číslo, zvýšime hodnotu premennej o zadané číslo)

 - zníženie hodnoty premennej A o 1

 - priradenie čísla 7 do premennej A

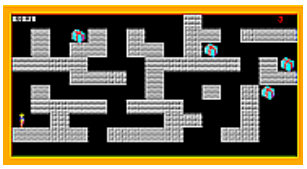
obrazovku na konkrétne súradnice, napr.  .

Pomocou prvku **písmo**  môžeme text sformátovať. Vložením tohto prvku do programu sa otvorí okno, v ktorom nastavíme písmo, jeho rez, veľkosť, farbu a pod. Príkaz potom vyzerá napr. takto:

Rovnakým spôsobom na obrazovku vypíšeme obsah premennej:

Úloha 2.12

Upravte program *bludisko.bpr* podľa zadania v *Bludisku 3*. Pomocou celočíselnej globálnej premennej sledujte počet pozbieraných balíčkov, obsah premennej priebežne vypisujte v pravom hornom rohu scény. Po pozbieraní všetkých balíčkov vypíšte na obrazovku nápis KONIEC a po stlačení klávesu alebo tlačidla myši nech program skončí.



Bludisko 4

Pribudlo časové obmedzenie. Na hru je vymedzených 60 sekúnd (ak chcete hru sťažiť, zmenšite časový limit). Ak sa balíčky nestihnú pozbierať, hráč prehráva (zobrazí sa príslušný nápis), ak stihne, vyhráva. V tejto verzii sa už využíva cyklus **while** a práca s prvkom **stopky**. Bežiaci čas hry v sekundách sa zobrazuje v ľavom hornom rohu scény. Baltík je premenený na chlapca.

V e-learningovom kurze sú pripravené všetky štyri programy s rôznym stupňom úrovne naprogramovania hry: *bludisko1.bpr, bludisko2.bpr, bludisko3.bpr, bludisko4.bpr.*

Úloha 2.13

Riešenie verzie 4 je pripravené v e-learningovom kurze pod názvom *bludisko4.bpr*. Pozrite si ho a vyskúšajte. Skúste v ňom zmeniť časový limit, a tým ovplyvniť náročnosť hry. Skúste namiesto „zbierania“ balíčkov balíčky „rozbaľovať“, t.j. vyčarovať namiesto balíčka nejaký predmet.

Animácie

Úloha 2.14

Baltík má sadu príkazov na tvorbu tzv. automatických animácií. V e-learningovom kurze je pripravený program *animácie.bpr*. Otvorte ho a pozrite si, ako sa v Baltíkovi animácie vytvárajú a ako vyzerajú, keď program spustíte.

Tvorivá autorská súťaž v programovaní **Creative Baltie** (pôvodný názov B+B) má dlhoročnú tradíciu. Pozrite si niektoré práce žiakov z minulých ročníkov - sú v e-learningovom kurze v priečinku *Práce žiakov*. Sú zbalené, pretože každý projekt obsahuje veľké množstvo súborov. Po rozbalení spustite súbor s príponou *.bpr*. Súťažné práce žiakov sú archivované a je možné ich stiahnuť na http://souteze.sgpsys.com/

Čo sme sa naučili

- orientovať sa v prostredí programovacieho jazyka Baltík 3, vedieť, aké možnosti výučby programovania poskytuje,
- pracovať v režimoch Skladať scénu a Čarovať scénu,
- ovládať základné príkazy z režimu Programovať - začiatočník, vedieť vytvoriť jednoduché programy,
- mať prehľad o viacerých príkazoch z režimu Programovať - pokročilý a o spôsobe ich používania.

3 IzyLogo

V tejto časti materiálu predstavíme mikrosvet, ktorý využíva základné myšlienky jazyka Logo, pomocou ktorého sa bližšie oboznámime so základnými princípmi programovania. Prostredie môžeme použiť aj priamo vo vyučovaní informatiky na druhom stupni ZŠ.

Prostredie umožňuje riadiť objekt v priamom režime, ale aj zostavovať vlastný program pomocou pripravených kartičiek. Na tvare objektu pritom vlastne nezáleží (niekedy ho budeme nazývať korytnačka, inokedy to bude autíčko, či včela). IzyLogo dovolí tiež pripravovať a využívať **vlastné príkazy**, t.j. časti programov, ktoré riešia iba nejakú časť zložitejšieho zadania.

Mikrosvet umožňuje:

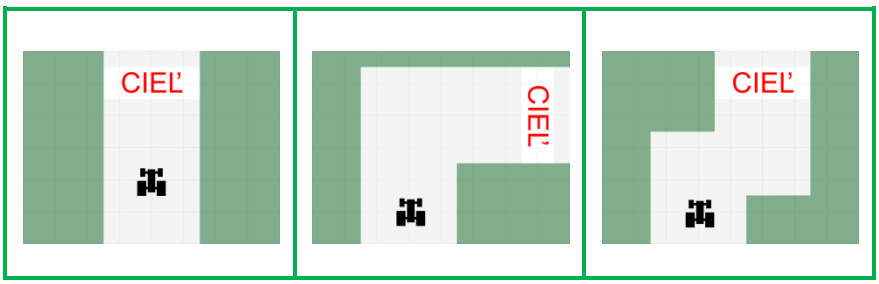
- riešiť sadu pripravených gradovaných úloh, ktoré sa postupne zobrazujú priamo v prostredí,
- riadiť jeden objekt – jeho pohyb do mrežových bodov a jeho otáčanie o násobky uhla 45°,
- vytvárať program na riadenie objektu s využitím základných príkazov a aj vytvorením vlastných príkazov.

V prostredí nájdeme príkazy-kartičky:

- na pohyb: dopredu, vľavo, vpravo,
- na nastavenie pera: nastav farbu, nastav hrúbku,
- pre vyplnenie nakreslenej oblasti,
- pre určenie náhodných vstupov pri kreslení a nastavení farby či hrúbky pera,
- na opakovanie skupiny príkazov (cyklus),
- na definovanie vlastných príkazov.

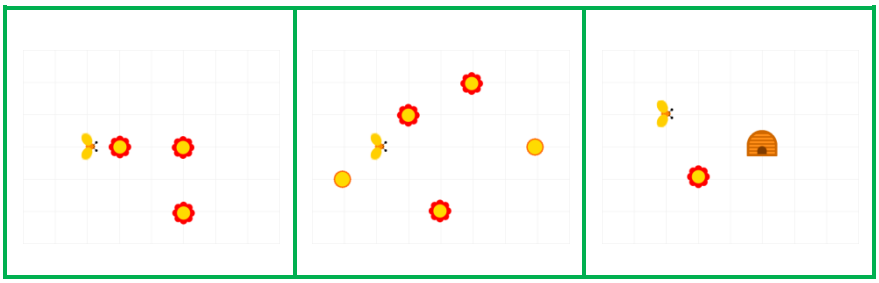
Pohyb v priamom režime

Klikaním na ikony s príkazmi v pravom páse vyriešte úlohy, ktoré sú pripravené v rámci prostredia. Autíčko sa pri klikaní na príkazy otáča alebo pohybuje podľa zvoleného príkazu vždy do nasledujúceho mrežového bodu.



Príprava programu

V nasledujúcich aktivitách je vašou úlohou prejsť včielkou cez všetky kvietky, ktoré sú na lúke. Nie však v priamom režime ako v minulých aktivitách, ale tak, že pripravíte **program**, ktorým včielka prejde a opelí všetky kvietky. Ak je na lúke aj úľ, včielka má skončiť svoje putovanie v ňom.



Program *IzyLogo.exe* nájdete v prostredí Moodle.

Pás s príkazmi

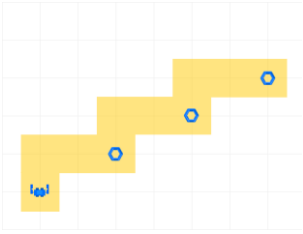
V prostredí budete riešiť aj ďalšie úlohy ako sú tie, ktoré uvádzame v študijnom materiáli. Pomocou nich si precvičíte a pochopíte jednotlivé príkazy.

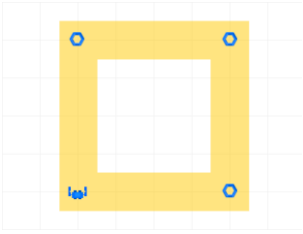
Sledujte, že autíčko prechádza vždy do nasledujúceho mrežového bodu nakreslenej štvorčekovej siete.

Štvorčeky plochy, a teda aj celý obrázok, môžeme zväčšovať kolieskom myši.

Základné príkazy, z ktorých pripravujeme cestovanie včely.

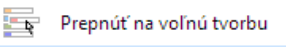
Pomocou ovládacieho prvku číselník (šípky pri jednotlivých kartičkách) môžeme zmeniť počet krokov alebo počet opakovaní.





Úloha 3.1	Riešte pripravené úlohy, ktoré ponúka prostredie. Pri riešení zostavujte krátke programy. Sledujte pohyb a otáčanie včielky. Využívajte základné príkazy na pohyb a otáčanie vľavo alebo vpravo o 45° alebo 90°.
Úloha 3.2	V nasledujúcich úlohách s robotom využite príkaz **opakuj**, ktorý pribudol v páse príkazov. V úlohách najprv vyhľadajte časti, ktoré sa opakujú. Potom navrhnite príkazy na prechádzanie robota po vyznačenej cestičke.
Úloha 3.3	Riešte aj ďalšie úlohy, v ktorých je potrebné opraviť už hotové postupnosti príkazov. Ťahaním príkazov z ľavej časti nahraďte niektoré kartičky v pripravenej postupnosti tak, aby program riešil pripravenú úlohu.

Prepnite prostredie na voľnú tvorbu.

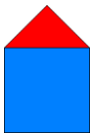


Tlačidlo v pravej časti na prepnutie prostredia na voľnú tvorbu príkazov.

Návrh a príprava vlastných príkazov

Ak nakreslíme niektorý obrázok, môžeme si ho uchovať ako svoj vlastný príkaz, ktorý budeme môcť využiť pri kreslení ďalších, zložitejších obrázkov.

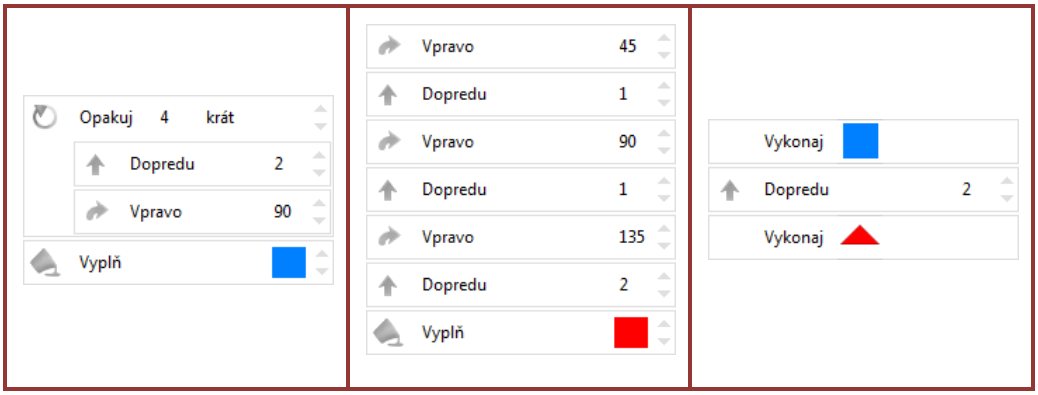
Navrhnime domček. Uvažujme najprv o dvoch útvaroch, z ktorých sa domček skladá – modrý štvorček a červený trojuholník.



Domček

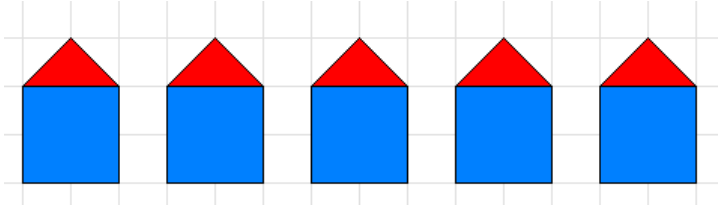
Najprv nakreslime modrý štvorček. Stlačením tlačidla + v pravom páse uložme tento obrázok ako nový príkaz. Ďalej navrhnime príkazy, ktorými sa kreslí červený trojuholník. Príkaz opäť uložme. Využime oba príkazy pri kreslení domčeka.

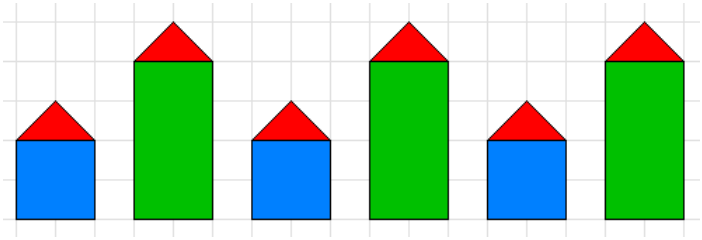
Nové príkazy sa objavia v pravom páse



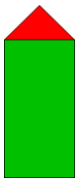
Aj príkaz na kreslenie domčeka si opäť uložme medzi nové príkazy, aby sme ho mohli použiť na nakreslenie ulice.

Pri navrhovaní každého príkazu si uvedomte, ktorým smerom je na začiatku korytnačka natočená.

Úloha 3.4	Nakreslite ulicu z piatich vedľa seba stojacich domov. 

Úloha 3.5	Podobne ako domček pripravte panelák – vyšší domček. Najprv pripravte zelený obdĺžnik. Na nakreslenie strechy nezabudnite využiť už pripravený červený trojuholník. Aj tento príkaz si uložte medzi nové príkazy.
Úloha 3.6	Kombináciou príkazov na kreslenie domčeka a paneláku nakreslite takúto ulicu. 

Uvedomte si, že je dôležité, kde a ako natočená skončí korytnačka kreslenie každého útvaru.



Panelák

Kde by sa malo kreslenie domčeka a paneláku skončiť, aby bolo kreslenie ulice čo najjednoduchšie?

Príprava vlastných aktivít

Úlohy, ktoré sa zobrazujú v režime Aktivít môžeme zmeniť a pripraviť svoju vlastnú sériu úloh. V priečinku, v ktorom je nainštalované IzyLogo, je podpriečinok *Aktivity*. V ňom sú textové súbory a ku každému z nich grafický súbor s rovnakým menom.

Tlačidlá na prechádzanie pripravenými aktivitami.

Aby ste videli, ktoré súbory patria k sebe, utrieďte priečinok *Aktivity* podľa mena.

Otvorme textový súbor *00.txt* v niektorom textovom editore. Preskúmajme jednotlivé riadky, uvažujme o tom, čo znamenajú:

Na otvorenie textového súboru použite napr. program *Poznámkový blok*.

Prezrite si aj iné textové súbory v tomto priečinku.

```
zadanie=1. Aktivita: Ako používať tento program - zvoľ si ďalšiu
aktivitu
rezim=priamy
pozadie=*.*
x0=439
y0=308
mriezka=64
objekt=auto.png
hrubka pera=1
farba pera=siva
smer=0
```

Každý riadok textového súboru začína informáciou o tom, ktorú **vlastnosť prostredia** určuje. Za názvom vlastnosti prostredia nasleduje znamienko „=“ (rovná sa). Môžeme určiť nasledujúce vlastnosti: zadanie, režim, pozadie, x-ová a y-ová súradnica objektu, mriežka, objekt, hrúbka a farba pera a smer objektu, ako je otočený pri štarte aktivity.

Pomocou textových súborov určujeme jednotlivé zadania. Súbory majú pevne stanovenú štruktúru:

Zadanie určuje textovú informáciu pre používateľa, ktorá sa objaví v hornom riadku. Môže mať maximálne 80 znakov.

Režim určuje, či bude prostredie umožňovať len pohybovať objektom, alebo aj vytvárať príkazy. Režim môže byť:

- priamy,
- jednoduchý,
- procedurálny.

Ak chceme **pripraviť vlastné aktivity**, musíme pripraviť a vytvoriť:

- **textový súbor** so zadaním úlohy,
- **obrázkový súbor** k úlohe,
- uvažovať o tom, ktorý **objekt** v aktivite použijeme. Pripravené sú súbory *auto.png*, *vcielka.png*, *robot.png* a *dievcatko.png*, a tiež objekt v tvare zeleného trojuholníka. V niektorom grafickom editore môžeme vytvoriť aj vlastný obrázok.

Úloha 3.7	Navrhnite aktivitu, ktorá v obrázku predkreslí schody a úlohou v zadaní bude naprogramovať pohyb objektu tak, aby prešiel po nich.

Ak je **pozadie** určené dvojicou `*.*`, prostredie využije obrázkový súbor s rovnakým názvom ako je názov textového súboru (bez prípony) a pridá k nemu príponu `png`.

Mriežka označuje veľkosť štvorčekov pri štarte hry.

Súradnice určujú, umiestnenie objektu na začiatku hry. Mriežka, ktorá je pripravená má veľkosť 64x64 bodov.

Objekt je názov obrázkového súboru aj s príponou.
Tento obrázok musí byť súčasťou priečinka Aktivity.

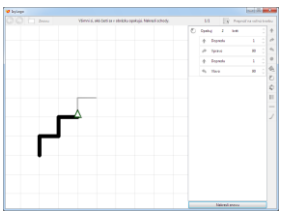
Nezabudnite, že meníme len nastavenia za znakom „=“ (rovná sa).

Všimnite si, že sme vynechali vlastnosť **objekt**. V prípade, že chcete použiť niektorý iný objekt ako trojuholník, môžete v textovom súbore túto vlastnosť uviesť a zapísať súbor, ktorý chcete využiť.

Jednoduchý režim znamená, že sa bude programovať pomocou kartičiek.

Pozadie - využije sa obrázkový súbor s rovnakým názvom ako je názov textového súboru.

Aby sme novú aktivitu nemuseli dlho hľadať medzi tými, ktoré boli už predtým pripravené v prostredí, presuňme pôvodné súbory napr. do priečinku *Aktivity_povodne*, ktorý na tento účel vytvoríme.

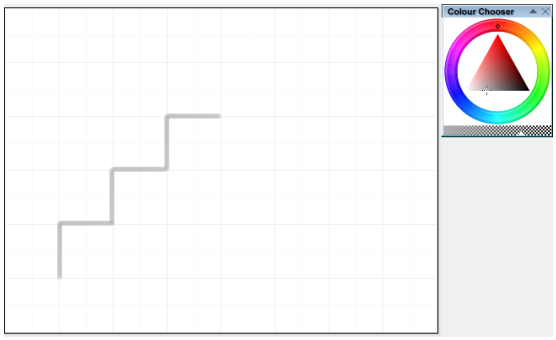


Naša aktivita v prostredí IzyLogo

Riešenie

Musíme pripraviť **obrázkový** a **textový** súbor k tejto aktivite. Obrázkový súbor pripravíme v obrázkovom editore RNA.

Otvorme súbor *mriezky.rna*, v ktorom je pripravená štvorčeková sieť. Budeme kresliť do poslednej fázy obrázka. Navrhnime napr. aktivitu na kreslenie schodov:



Nakreslime obrázok Aktivity.

Ak máme obrázok so zadaním pripravený, zmažme prvé dve fázy v súbore a obrázok zapíšme do súboru, napr. *01schody.png*.

Textový súbor pripravíme najjednoduchšie tak, že otvoríme iný existujúci súbor a pre prepíšeme v ňom niektoré údaje.

```
zadanie= Nakresli schody.
rezim=jednoduchy
pozadie=*.*
x0=64
y0=320
mriezka=64
hrubka pera=1
farba pera=cierna
smer=0
```

Súradnice `x0` a `y0` určujú počiatočnú polohu objektu. Závisia od veľkosti mriežky, ako sme ju navrhli v obrázkovom súbore - v našom prípade 64 bodov. Pri ich určovaní si treba uvedomiť, kde začína naša kresba, a tam umiestniť objekt. Schody začínajú v mrežovom bode posunutom vo vodorovnom smere o jeden štvorček vpravo od ľavého okraja, `x0=64`. Hodnotu `y0` určíme tak, že spočítame počet mrežových bodov od horného okraja v zvislom smere a vynásobíme ho číslom 64, `y0` je teda 5x64, t.j. 320.

Takto pripravený súbor zapíšme ako *01schody.txt*.

Úloha 3.8

Rovnakým spôsobom pripravte ďalšie aktivity.

Čo sme sa naučili

Spoznali sme program, ktorý je vhodný pre žiakov druhého stupňa ZŠ na oboznámenie sa so základmi programovania.

Dokážeme tiež pripraviť svoju vlastnú postupnosť úloh v prostredí IzyLogo.

4 Živý obraz

Program *zivyObraz.exe* nájdete v prostredí Moodle.

V nasledujúcej časti materiálu spoznáme prostredie Živý obraz, v ktorom budeme pripravovať rôzne aktivity. Pomocou nich sa oboznámime so základmi objektovo-orientovaného programovania. Spoznáme, že v prostredí môže byť niekoľko objektov. Každý z nich môže mať určené vlastnosti a svoje vlastné správanie v prípade, že nastane nejaká udalosť (napr. priKliknutí, priZrážke). Pomocou tohto prostredia ľahšie pochopíme, že objekty sa môžu hýbať aj naraz (paralelizmus).

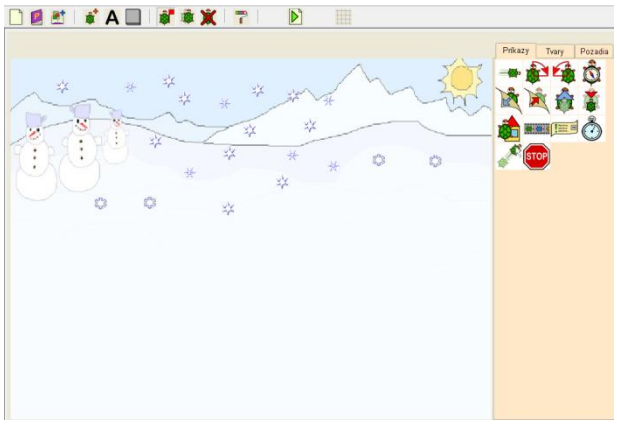
Takýmto spôsobom vytvoríme

- **neinteraktívne animácie**, teda také, v ktorých sa budú objekty pohybovať istým spôsobom, ich činnosť nedokážeme ovplyvniť,
- **interaktívne animácie**, t.j. niektoré objekty v obraze budú reagovať na kliknutie, prípadne na zrážku s inými objektmi,
- **skladačky**, **labyrinty**, **preteky**.

Prostredie

V prostredí si všimnime nasledujúce časti

- pracovná plocha,
- vodorovný pás tlačidiel,
- tri záložky Príkazy, Tvary a Pozadia v pravej časti. Každá záložka má svoj vlastný obsah.



Obr. 4.1 Prostredie programu Živý obraz

V pracovnej ploche môžeme meniť pozadie a vkladať do nej objekty.

Objekty môžeme riadiť priamo alebo programom pomocou príkazov zo záložky *Príkazy*.

Úloha 4.1

Navrhnite nejaké pozadie. Vložte do neho niekoľko objektov. Zmeňte objektom tvar. Smer každého objektu zmeňte tak, aby vyjadroval jeho skutočné natočenie.

Vyskúšajte pohyb každého objektu.

Vloženie pozadia

- zvolíme záložku **Pozadia** a vyberieme vhodné pozadie,
- kliknutím na obrázok v záložke sa tento obrázok vloží ako pozadie.

Vloženie objektu

- v hlavnej ponuke zvolíme **tlačidlo na vkladanie objektu**,
- kliknutím do pozadia vložíme nový objekt, ktorý má zatiaľ tvar korytnačky.

Tvar objektu

- zvolíme záložku **Tvary**, klikneme na niektorý z ponúkaných tvarov, objekt zmení tvar,
- pozor, ak máme na ploche niekoľko objektov, mení sa tvar toho, na **ktorý sme naposledy klikli**.

Natočenie objektu

- kliknutím na objekt vyberieme objekt, ktorého smer chceme meniť,

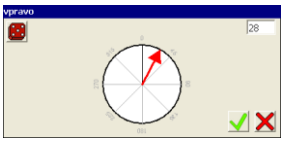
Objektom môžeme vyberať tvary zo záložky *Tvary*.

Kliknutím na obrázok pozadia sa tento obrázok vloží ako pozadie pracovnej plochy.

Tlačidlo na vkladanie objektu

Ikona príkazu na zmenu smeru

- v záložke **Príkazy** vyberme **príkaz na zmenu smeru**,
- uhol natočenia vyberáme v otvorenej **pomôcke** kliknutím do kruhu alebo tak, že priamo napíšeme nejaký uhol,
- natočenie sa udáva v stupňoch, 0 určuje smer hore, ostatné smery sú tak, ako sme zvyknutí z klasickej geometrie.

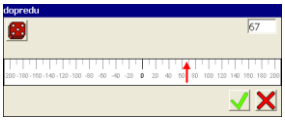


Pomôcka na výber uhla

Pohyb objektu

- klikneme na objekt, ktorým chceme pohnúť,
- v záložke **Príkazy** vyberieme **príkaz pohybu**,
- otvorí sa pomôcka, ktorou môžeme určiť, ako veľký krok má objekt urobiť,
- vzdialenosť vyberáme kliknutím do pravítka alebo zapísaním čísla.

Ikona príkazu pohyb



Pomôcka na výber vzdialenosti

Rôzne typy projektov v prostredí Živý obraz

Neinteraktívne a interaktívne animácie

Na internete sa čoraz častejšie objavujú obrázky, na ktorých sa niečo hýbe. Môžeme ich rozdeliť do dvoch kategórií: **neinteraktívne**, t.j. pohyb sa nedá ničím ovplyvniť, alebo **interaktívne**, t.j. na niektoré objekty na obrázku sa dá kliknúť, pričom ich správanie sa vtedy zmení. V prostredí *Živý obraz* sa dajú vytvárať oba typy takýchto animácií.

Úloha 4.2

Navrhnite neinteraktívnu animáciu, v ktorej bude padať sneh.

Popis aktivity a riešenie

Do pozadia plochy vložíme obrázok. Keďže budeme uvažovať o snežení, vhodná je zimná krajinka so snehuliakmi. Ostatné objekty na ploche budú vločky, ktoré sa pri spustení živého obrazu budú pohybovať. Tie nepatria pozadiu, pretože majú v rámci živého obrazu nejaké správanie – pohyb smerom dolu.

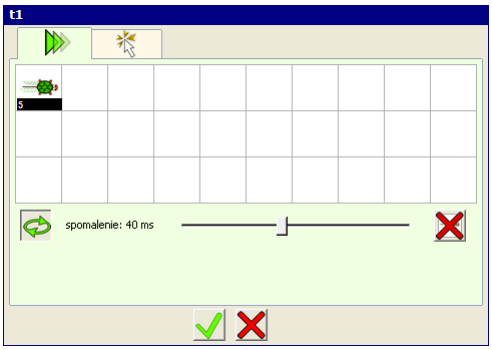
Vložíme pozadie a jeden objekt, ktorému vyberieme tvar vločky. Pripravme vločke správanie v tomto živom obraze.

Uvedomte si, že všetky vločky budú mať rovnaké správanie – budú natočené smerom nadol a po spustení živého obrazu budú padať smerom dolu.

Správanie objektu

- kliknime na objekt **pravým tlačidlom**, otvorí sa nové okno, ktoré budeme nazývať **okno správania objektu**,

Okno správania objektu má niekoľko záložiek. Zatiaľ využívame záložku pri spustení, t.j. navrhujeme správanie objektu, ktoré sa bude vykonávať hneď po spustení živého obrazu.



Obr. 4.2 Okno správania objektu

Ak omylom vyberieme nesprávny príkaz kliknime naň v okne správania objektu pravým tlačidlom – príkaz sa zruší.

- v záložke **Príkazy** vyberme **príkaz pohybu,**
- v otvorenej **pomôcke** zvoľme dĺžku, o ktorú sa má vločka pohnúť,
- zatvorme okno výberom tlačidla  .

Tlačidlá na spustenie a zastavenie živého obrazu

Spustenie živého obrazu

- vyskúšajme projekt tlačidlom na **spustenie živého obrazu**,
- po spustení sa vločka pohne o určený počet krokov,

- zastavme projekt tlačidlom na **zastavenie živého obrazu**.

Nekonečný pohyb

- otvorme opäť **okno správania objektu**,
- stlačme tlačidlo donekonečna  - príkazy, ktoré sú určené v okne sa budú vykonávať stále, až kým nezastavíme živý obraz,
- určme spomalenie – ťahajme posúvač.

Tlačidlo na klonovanie objektu

Klonovanie objektu

- ak projekt s jednou vločkou funguje správne, vločku naklonujeme na ďalšie miesta,
- vyberme tlačidlo na klonovanie objektu,
- kliknime na objekt, ktorý chceme klonovať,
- kliknime na nové miesto, kam chceme objekt s rovnakým správaním umiestniť,
- vyskúšajme projekt,
- uložme projekt.

Ak pri stlačení tlačidla na klonovanie zároveň stlačíte tlačidlo **Ctrl**, môžete vkladať klony, až kým znovu nestlačíte tlačidlo na klonovanie – takto môžete vložiť veľa vločiek bez toho, aby ste znovu museli určiť, ktorý objekt chcete klonovať.

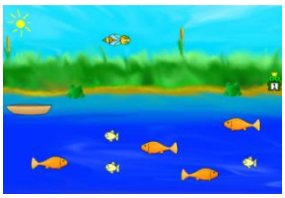
Tlačidlo na uloženie projektu

Úloha 4.3

Do projektu vložte toľko vločiek, koľko potrebujete. Ďalej upravte projekt tak, aby sa niektoré vločky pohybovali pomalšie, iné rýchlejšie.

Úloha 4.4

Navrhnite živý obraz, v ktorom budú v rybníku plávať rybky, skákať žaby, lietať balón, atď. Projekt pripravte tak, aby sa rybka pri kliknutí na ňu otočila, aby žabka pri kliknutí poskočila.



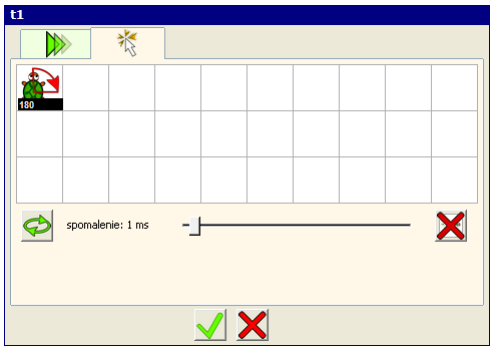
Rybník s plávajúcimi rybkami a vyskakujúcimi žabkami

Riešenie

Pozadie a **jeden objekt s tvarom rybky** vložíme rovnakým spôsobom ako v predchádzajúcom projekte.

Pri kliknutí na rybku potrebujeme, aby sa otočila do „protismeru“, t.j. o 180 stupňov.

Tie príkazy, ktoré má objekt vykonať pri kliknutí naň, vkladáme tiež v **okne správania**, avšak v záložke **pri kliknutí**:



Záložka pri kliknutí s príkazmi, ktoré objekt vykoná, keď naň klikneme

Všimnite si, že po zastavení živého obrazu sa objekty vrátili na svoje pôvodné miesto.

Ak chceme zrušiť niektorý objekt, vyberme tlačidlo na rušenie objektov a potom kliknime na objekt.

Tlačidlo na zrušenie objektu

Úloha 4.5

Navrhnite iné interaktívne animácie.

Uvažujte napríklad pozadie s cestou, na ktorej sú domy a po ktorej chodia autá, bicykle a pes.

Skladačky

V prostredí Živý obraz sa dajú jednoducho vytvárať aj rôzne typy skladačiek. Pomocou príkazov na zmenu záberu vybraného objektu, či skrývanie objektov si môžeme vytvoriť vlastný avatar (postavičku, ktorá nás bude reprezentovať v počítačovom svete) alebo prostredie na hádanie toho, čo je na obrázku, či iných skladačiek, napr. tangram, sudoku.

Môj avatar

Ikona príkazu ďalší záber

Vhodným výberom štvorčekov dokáže hráč rýchlo uhádnuť, čo je na obrázku.

Najprv pripravte jeden štvorček s vhodným správaním (pri kliknutí sa štvorček skryje). Potom využite klonovanie, nezabudnite využiť kláves **Ctrl**.

Ikona príkazu skry objekt.

Pri vytváraní hry využite sieť – štvorčeky, ktoré zakrývajú objekt budú pekne vyrovnané.

Tlačidlo sieť a posúvač, ktorým sa dá vybrať hustota siete

Nezabudnite, že najvýhodnejšie je použiť klonovanie objektu, ktorý už má pripravené správanie.

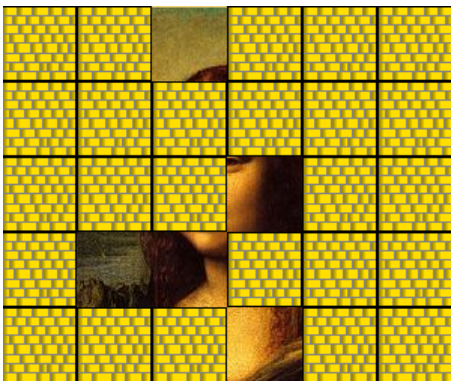
Uvedomte si, že niekedy je správanie objektu dôležitejšie ako obrázok, ktorý objekt zobrazuje.

Úloha 4.6

Navrhnite prostredie, v ktorom si budeme môcť skladať svoju vlastnú tvár–avatara. Pomocou klikania na vlasy, oči, nos, ústa a tvár sa bude meniť ich tvar.

Úloha 4.7

Poznáte hru Hádaj, kto je to?

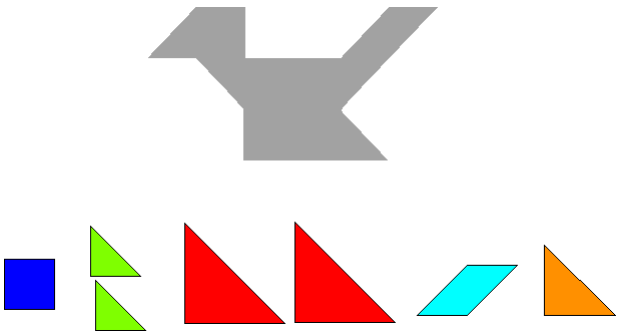


Hádaj, kto je to?

Pozadie hry je fotka známej osobnosti, obraz, nejaký predmet a pod. Táto fotka je zakrytá, napr. niekoľkými štvorčekmi. Úlohou hráča je uhádnuť, kto je na obrázku odkrytím čo najmenšieho počtu štvorčekov. Keď hráč klikne na niektorý štvorček, tento sa mu odkryje, aby videl viac z fotky.

Úloha 4.8

Navrhnite skladačku, tzv. tangram.



Tangram

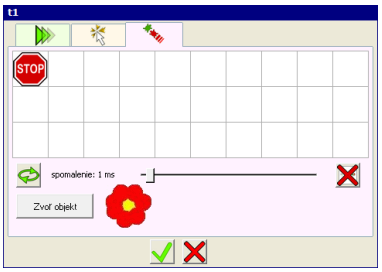
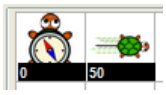
Objekty z dolnej časti sa dajú ťahať do predpripraveného pozadia, na ktorom je tieň hotového tangramu.

Pri klikaní na objekt sa objekt otočí – vyberie si ďalší záber.

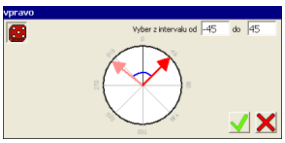
Zrážky

Ďalší typ aktivít, ktoré môžeme robiť v prostredí Živý obraz, sú také, v ktorých potrebujeme zistiť, či sa niektoré objekty stretli, t.j. narazili na seba.

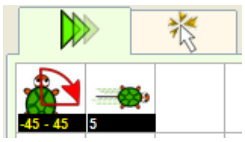
Kvietky, ktoré sú v zadnej časti obrázka môžete zmenšiť, aby sa zachovala perspektíva.

Úloha 4.9	Navrhnime takýto projekt: včielka lieta po lúke a keď nájde niektorý červený kvietok, zastane.
Analýza riešenia	**Pozadie** a **objekty** s obrázkami kvietkov pripravme rovnako, ako sme to robili v predchádzajúcich projektoch. Pripravme aj objekt s obrázkom včielky. Uvažujme, čo potrebujeme ďalej: • včielka sa musí neustále pohybovať, • keď sa stretne s niektorým kvietkom, zastane. Pohyb včielky navrhneme podobne ako v predchádzajúcich projektoch, nezabudnime zvoliť nekonečný pohyb. Ďalej potrebujeme zistiť, či včielka našla niektorý kvietok. Na to využijeme v okne správania včielky záložku **pri zrážke**. Stlačením tlačidla **Zvoľ objekt** a kliknutím na červený kvietok vyberieme jej správanie v prípade, ak sa stretne s hociktorým červeným kvietkom:  *Záložka pri zrážke včielky s kvetom*
Úloha 4.10	Navrhnite projekt, v ktorom môže hráč prechádzať labyrintom. V programe kontrolujte, či sa panáčik nezrazil so stenou a tiež, či našiel poklad.
Nápoveď	Pre pohyb panáčika **smerom hore** pomocou tlačidla vložíme v okne správania tlačidla ikonický program: 

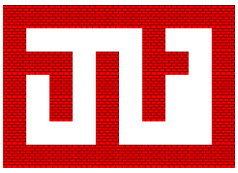
Ikona príkazu zmenši objekt.



Pomôcka pre otočenie o náhodný uhol



Pohyb včielky



Jednoduchý labyrint, ktorý pripravíme klonovaním jednej steny

Tlačidlo vkladania nových tlačidiel do projektu

Pozor: tlačidlom sa bude ovládať objekt, na ktorý sme klikli **tesne predtým**, ako sme spustili živý obraz.

Ak chceme, obrázok na tlačidle môžeme zmeniť rovnako ako pre ľubovoľný objekt – v záložke Tvary kliknime na vhodný tvar, napr. šípku.

Čo sme sa naučili

Spoznali sme program Živý obraz, ktorý sa dá využiť pre žiakov druhého stupňa ZŠ na oboznámenie sa so základnými princípmi programovania – postupnosť príkazov, cyklus, objekty a ich riadenie pomocou udalostí: kliknutie myšou, zrážka s iným objektom.

5 Scratch

Scratch je bezplatný softvér lokalizovaný aj do slovenského jazyka. Inštalačný program je možné získať na stránke http://info.scratch.mit.edu/Scratch_1.4_Download

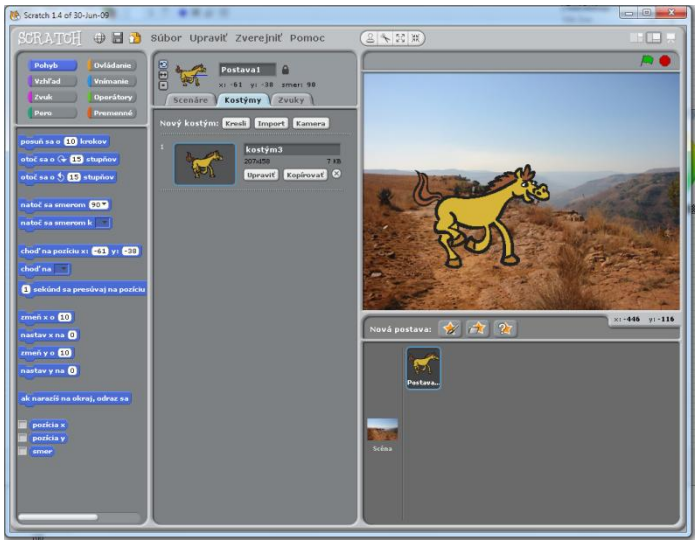
Ďalšie zdroje užitočné pri tvorbe projektov – obrázky, zvuky, programy, pozadia môžeme stiahnuť zo stránky http://resources.scratchr.org/resources

Scratch je prostredie vhodné aj na vyučovanie programovania na 2. stupni ZŠ. Dajú sa v ňom ľahko vytvárať interaktívne príbehy, animácie, hry a žiaci ich môžu tiež zdieľať na celosvetovej webovej stránke. Bol vytvorený skupinou Lifelong Kindergarten Group pod vedením Mitchela Resnicka na Massachusettskom technologickom inštitúte MIT. Autori uvádzajú tri základné princípy, ktoré sledovali pri návrhu programovacieho prostredia [11]:

- **Hravosť** – programovanie v Scratchi je inšpirované stavebnicou Lego alebo skladačkou puzzle: program sa vytvára skladaním z blokov, ktoré do seba zapadajú len v prípade syntakticky správneho programu. Programovanie je interaktívne, žiak experimentuje so skladaním blokov programu, prirodzene vytvára viaceré paralelné skladačky, iteratívne zmeny sa dajú robiť aj počas behu programu.
- **Zmysluplnosť** – vyučovanie je efektívne, keď žiaci pracujú na zmysluplných projektoch reflektujúcich záujmy mladých ľudí: grafika, animácie, fotografie, hudba, zvuky. V Scratchi sa dajú vytvárať rozmanité projekty atraktívne pre mladých ľudí: interaktívne animované príbehy, videohry, interaktívne časopisy, narodeninové pohľadnice, tutoriály, vedecké simulácie, súťaže.
- **Sociálnosť** – prostredie Scratch má vstavanú funkciu zdieľania vytvoreného projektu na webovej stránke http://scratch.mit.edu/, kde je možné spustiť ho vo webovom prehliadači, okomentovať, ohodnotiť alebo stiahnuť program pre nahliadnutie, úpravu, doprogramovanie a pod. Tisícky programov vytvorených komunitou používateľov Scratchu sú bohatým zdrojom motivácie, inšpirácie, učenia sa.

Prostredie Scratch

Prostredie Scratch je rozdelené na štyri hlavné časti:



Obr. 5.1 Prostredie Scratch

Práca v prostredí pripomína prácu tvorcu filmu: príprava scény, postáv, ich kostýmov a scenárov pre jednotlivé postavy.

V ľavej časti prostredia sa nachádzajú **príkazy**, rozdelené do niekoľkých skupín: Pohyb, Vzhľad, Zvuk, Pero, Ovládanie,... Nimi môžeme riadiť postavy.

V strednej časti prostredia vytvárame pre postavu jej **Scenáre**, vyberáme pre ňu **Kostýmy** a pripravujeme **Zvuky**.

V pravej časti je **Scéna**. V jej dolnej časti je prehľad všetkých objektov v projekte – ikony scény a postáv na scéne.

Každá postava má **svoje vlastné** Scenáre, Kostýmy a Zvuky. Ak chceme vidieť **Scenáre** niektorej postavy, klikneme na jej ikonu v pravej dolnej časti.

Ak klikneme na záložku **Kostýmy**, uvidíme, do akých kostýmov sa môže daná postava prezliekať.

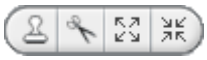
V záložke **Zvuky** môžeme pripraviť pre postavu tie zvukové súbory, ktoré budeme v projekte využívať.

Úloha 5.1	Zvoľte skupinu príkazov **Pohyb** a klikajte na niektoré príkazy. Sledujte reakcie mačky, ktorá je v prostredí zvolená ako základná postava.
Úloha 5.2	Pripravte iný obrázok scény, postavu oblečte do kostýmu koňa. Vyskúšajte postave aj iné kostýmy.

Riešenie	Zvoľte záložku **Kostýmy** a pomocou tlačidla **Import** načítajte pre postavu jej nový kostým. Kliknite na tento kostým – postava sa do neho prezlečie.
Úloha 5.3	Pridajte do scény nové postavy a vyskúšajte si ich riadenie klikaním na príkazy v ľavej časti prostredia. Vyskúšajte si tiež vytváranie duplikátov, rušenie postáv, ich zväčšovanie a zmenšovanie.
Úloha 5.4	Pripravte nejakú scénu – zvoľte pozadie scény, umiestnite na ňu niekoľko postáv. Riaďte každú postavu pomocou klikania na príkazy.

Na pridávanie nových postáv využite tlačidlá:

Na vytváranie duplikátov, rušenie postáv, zväčšovanie a zmenšovanie využite nasledujúce tlačidlá



Postavu môžeme myšou ťahať po scéne a umiestňovať ju na rôzne miesta.

Všimnite si, čo sa deje s postavou, ak by chcela ísť za okraj scény.

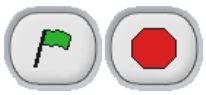
Postupnosť príkazov

V prechádzajúcich úlohách sme riadili jednotlivé postavy tak, že sme si kliknutím vybrali postavu a potom sme ju mohli riadiť príkazmi.

V prostredí Scratch však môžeme pre postavu pripraviť scenár, t.j. postupnosť príkazov, ktoré má vykonať. Scenár sa môže spustiť pri kliknutí na zelenú zástavku a vytvárame ho v strednej časti prostredia.

Úloha 5.5	Do scény pripravte jednu postavu. Umiestnite ju na ľavý okraj scény. Navrhnite pre ňu nasledujúci scenár, ktorý sa spustí pri kliknutí na zelenú zástavku: • nech sa postava presunie na pravú stranu scény, • nech niečo povie, napr. „Ahoj!“. Upravte scenár tak, aby postava prešla na pravú stranu scény nie jedným krokom, ale aby tam prechádzala postupne, napr. počas 2 sekúnd.
Riešenie	Na prípravu uvedeného scenáru použite príkazy: 
Úloha 5.6	Pridajte do scény niekoľko ďalších postáv, pre ktoré pripravíte ich vlastné scenáre, v ktorých sa budú prechádzať po scéne a niečo hovoriť.

Ak chceme, aby postavy začali vykonávať svoje scenáre pri kliknutí na zelenú zástavku, použijeme príkaz



Tlačidlá na spustenie a zastavenie scenárov.

Uvedomte si, že príkazy, ktoré sú zobrazené v strednej časti sú určené pre postavu, ktorej ikona je vyznačená v pravej dolnej časti.

Projekt si uložte – využite hlavnú ponuku **Súbor**.

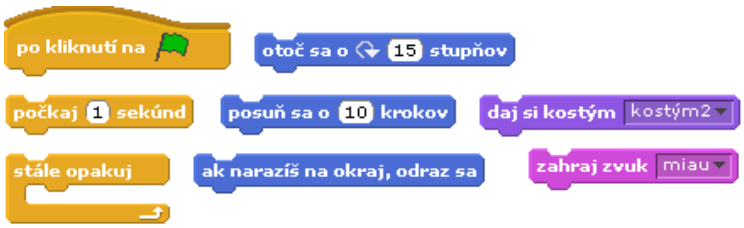
Opakovanie postupnosti príkazov

Úloha 5.7	Navrhnite nasledujúcu scénu a scenáre: • scéna má obrázok pod vodou, • na scéne vystupuje niekoľko postáv-rybičiek, • rybičky neustále plávajú pod vodou.

V hornej časti sa postavám dá zmeniť smer pohybu: ťahajte za modrú čiaru a otáčajte postavu

Tlačidlá na voľbu spôsobu otáčania a čiara na zmenu smeru postavy.

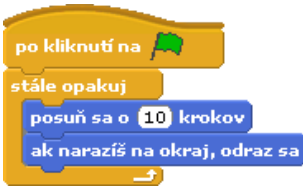
Postave sa dá tiež zmeniť spôsob správania pri jej otáčaní: niekedy je vhodné, aby sa obrázok otáčal, inokedy je lepšie, keď má obrázok len dva smery.

Riešenie	Pri príprave scenárov využite príkazy: 
Úloha 5.8	Prezrite si knižnicu obrázkov pre scénu a tiež knižnicu obrázkov pre postavy. Navrhnite ďalšie scenáre, v ktorých sa postavy pohybujú, niečo hovoria, alebo si prezliekajú kostýmy.
Nápoveď	Uvažujte napr. scény: na diskotéke, virtuálna pohľadnica z môjho mesta, o mojom dome a pod.

Podmienky

Pri vytváraní čo i len trochu zložitejších scenárov sa nevyhneme testovaniu toho, či nastala nejaká udalosť. Môžeme napríklad zistiť, či sa niektorá postava stretla s inou postavou, či sa dotýka nejakej farby, či bola stlačená myš alebo pomocou rôznych operátorov môžeme pripravovať aj zložitejšie výrazy.

Postavy môžeme premenovať, ak do textového poľa vedľa postavy v strednej časti prostredia zapíšeme namiesto slova postava jej nové meno:



Scenár pre mačku

Scenár pre psa

Úloha 5.9	Navrhnite nasledujúci scenár: • na lúke sú pes a mačka, • mačka sa neustále pohybuje nejakou rýchlosťou, na okrajoch sa odráža, • pes sa snaží ísť za mačkou, keď ju stretne, zahavká.
Riešenie	Podobne ako v predchádzajúcich úlohách, importujme pozadie scény, vytvorme dve postavy s obrázkami mačky a psa. Mačke pripravme nasledujúci scenár: • pri kliknutí na zelenú zástavku sa bude stále pohybovať krokom 10 a odrážať od okrajov. Psovi pripravme scenár: • pri kliknutí na zelenú zástavku sa bude stále pohybovať tak, že vyrazí smerom k mačke a posunie sa o niekoľko krokov. Vyskúšajme scenáre. Ešte potrebujeme, aby pes pri stretnutí s mačkou zareagoval štekaním. Využijeme na to základné príkazy  Doplňme ich do scenára pre psa tak, aby stále zisťoval, či sa dotýka mačky a ak áno, aby zaštekal.

Úloha 5.10	Doplňte do scény ďalšie postavy a navrhnite im ich vlastné scenáre.
Nápoveď	Uvažujte napr. aj o tom, že niektoré postavy sa pri stretnutí pozdravia, iné skryjú a potom objavia sa na inom mieste. Využite aj nasledujúce príkazy: 

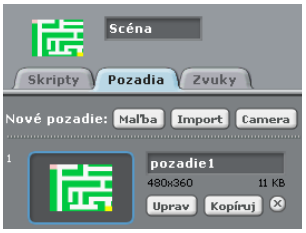
Do scény sme pridali postavu malého psíka. Uvažujte a popíšte slovami, aký scenár sme mu pripravili takouto postupnosťou príkazov:

Stláčanie klávesov

Prostredie Scratch podporuje riadenie postáv pomocou klávesov. Takýmto spôsobom sa dajú pripraviť aktivity, v ktorých ovládame postavu napr. pomocou šípok a táto putuje po nejakom mieste. Takto pripravené aktivity môžu byť labyrinty, v ktorých prechádzame z jedného miesta na druhé, mestá, po ktorých sa prechádzame a spoznávame ich uličky. Môžeme tiež pripraviť prechádzku s mimozemšťanmi a zažiť tak rôzne dobrodružstvá na ich planéte.

Labyrint môžeme nakresliť priamo v prostredí

Úloha 5.11	Nakreslite vlastný labyrint. Navrhnite postavu, zvoľte jej kostým a pripravte nasledujúci scenár pre ňu: • na začiatku sa umiestni do ľavého horného rohu labyrintu – štart, • bude reagovať na stláčanie klávesov – šípka hore, dolu, vpravo a vľavo tak, že prejde tým smerom niekoľko krokov, • ak narazí na stenu, vráti sa späť na štart, • keď príde do cieľa (môže byť vyznačený nejakou farbou), povie „Vyhral som!“
Riešenie	Obrázok labyrintu môžeme nakresliť priamo v prostredí Scratch – pre scénu vyberieme záložku pozadia a zvolíme tlačidlo Maľba. Pre steny labyrintu zvolíme jednu farbu. Do cieľa labyrintu umiestnime inú farbu. Uvažujme aj o tom, aby boli chodby labyrintu dosť široké. V prostredí umiestnime postavu do ľavého horného rohu labyrintu. Zmenšime ju tak, aby sa zmestila do chodieb. Pripravme pre ňu nasledujúci scenár: • pri stlačení šípky vpravo sa pohne o niekoľko krokov vpravo  *Ovládanie postavy pomocou šípky vpravo*



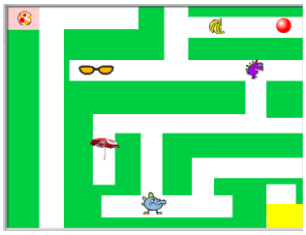
Uvažujte o dĺžke kroku, ktorý má postava urobiť, aby dokázala prejsť labyrintom.

Ak je potrebné zmenšite postavu, aby sa zmestila na chodby labyrintu.

Pri príprave príkazov využite ich kopírovanie a úpravu.

Nové postavy - pohyblivé steny

Pri príprave nových postáv môžete využiť kopírovanie príkazov a ich uloženie pre novú postavu.



Labyrint

Hotový labyrint si môžete uložiť na webové stránky Scratch.

	Keď projekt vyskúšame, všimnime si, že postava zatiaľ vstupuje aj do steny labyrintu. Pokračujme v príprave scenára • musíme navrhnúť testovanie, či sa postava nedotkla farby, ktorú majú steny labyrintu. Vyskúšajme stláčať šípku. Keď sa postava dotkne vybranej farby, vráti sa na štart. Podobným spôsobom pripravte scenár pre ovládanie postavy pomocou ostatných šípok.
Úloha 5.12	Doplňte scenár tak, aby postava zisťovala, či už nie je v cieli.
Riešenie	To, či sa postava nedotýka farby, ktorú sme si zvolili pre cieľ, môžeme kontrolovať dvoma spôsobmi: • do každého príkazu pohybu (t.j. do všetkých štyroch príkazov pri stlačení klávesu) pridáme príslušný test, • vytvoríme nový príkaz, ktorý bude stále kontrolovať, či už nie je v cieli.
Úloha 5.13	Do labyrintu vložte pohyblivé steny, t.j. postavy, ktoré budú vyzerať ako časť steny. Pripravte pre ne taký scenár, že sa budú zobrazovať a skrývať v nejakých časových intervaloch.
Úloha 5.14	Do labyrintu vložte nejaké ďalšie postavy, s ktorými sa postava stretne a pozdraví sa s nimi. Navrhnite scenáre pre tieto postavy.
Úloha 5.15	Do labyrintu vložte aj nejaké predmety, ktoré musí postava pozbierať – napr. ovocie, loptičky a pod.
Úloha 5.16*	Nakreslite ďalšie pozadie – labyrint. Doplňte správanie postavy, ktorá prechádza labyrintom tak, aby pri príchode do cieľa prešla na nové pozadie.
Riešenie	Pri riešení tejto úlohy si musíme uvedomiť • postava kontroluje steny podľa farby (na našom obrázku zelenej), t.j. aj steny nového pozadia musia byť zelené, • pri príchode do cieľa v jednej miestnosti labyrintu potrebujeme, aby si scéna nastavila ďalšie pozadie. Na takúto komunikáciu sa v prostredí Scratch používa posielanie správ: • zo skupiny **Ovládanie** vyberieme príkaz **pošli všetkým správu** a zapíšeme text správy, ktorú chceme poslať, napr. **dalsiePozadie**, • správu vložíme do skupiny príkazov, ktoré sa vykonajú v prípade, keď postava príde do cieľa, • scéne doplníme jej vlastný scenár, v ktorom zo skupiny Ovládanie vyberieme **po obdržaní správy** a zo skupiny **Vzhľad** vyberieme **nasledujúce pozadie**.

Premenné

Dôležitou súčasťou programovacích jazykov sú premenné. Väčšina programovacích jazykov podporuje prácu s premennými. Medzi malými programovacími jazykmi však nájdeme aj také, ktoré neobsahujú premenné (napr. Karel).

V prostredí Scratch je na prácu s premennými pripravená skupina **Premenné**. V nej môžeme vytvárať svoje vlastné premenné, ktorým nastavujeme, či pôjde o premennú pre jednu alebo pre všetky postavy.

V projektoch, ktoré sú zaujímavé pre žiakov na 2. stupni ZŠ, budeme premenné najčastejšie využívať na počítanie toho, koľkokrát sa stala nejaká udalosť (napr. koľkokrát sme klikli na niektorú postavu) alebo na počítanie času.

Nakreslite si Mandalu: http://scratch.mit.edu/projects/beetle16/1009497

Koncepty:
- premenné
- náhodné vstupy

Príkazy:

Postupy:
- viacnásobné klonovanie objektov aj s ich scenármi

Príkaz, ktorým počítač vyberie náhodné číslo.

Použijeme ho pri generovaní x-ovej, y-ovej súradnice pri štarte programu, ale aj pri nastavovaní farby efektu, či otáčaní sa o náhodný uhol.

Úloha 5.17

Pripravte projekt-hru:
- na hracej ploche je niekoľko lentiliek a jedna z nich je tmavosivá,
- lentilky sa náhodne umiestnia na plochu a potom sa pohybujú – trasú,
- úlohou hráča je kliknúť na tmavosivú lentilku,
- program spočítava, koľkokrát sa hráčovi podarilo kliknúť na sivú lentilku,
- zároveň sa hráčovi počíta čas, ako dlho hru hral.

Riešenie

Postupne pripravme
- postavu s obrázkom tmavosivej lentilky,
- jej scenár, v ktorom sa pri štarte presunie na náhodnú pozíciu, a potom sa bude stále otáčať do náhodného smeru, posúvať o 3 kroky,
- ďalšiu lentilku, ktorá sa bude od tmavosivej líšiť farbou a v scenári tým, že si pri štarte nastaví farbu efektu na náhodné číslo (takto získame rôznofarebné lentilky).

Vyskúšajme program pre tieto dve lentilky.

Kopírovaním zelenej lentilky pomocou pečiatky (pričom zároveň držíme tlačidlo **Shift**) vložme na scénu niekoľko ďalších lentiliek. Projekt opäť vyskúšajme.

Doplňme do programu premennú na počítanie – v skupine **Premenné** zvoľme *Vytvoriť premennú* a pomenujme ju **skore**.

Pre tmavosivú lentilku doplňme scenár o udalosť **po kliknutí na lentilka1** tak, aby sa skóre zvýšilo o 1.

Vytvorme aj ďalšiu premennú **cas**, pre ktorú budeme každú sekundu opakovať jej zvyšovanie.

Aby bola hra zaujímavejšia, môžeme v nej doplniť presunutie sivej lentilky po kliknutí na ňu na náhodnú pozíciu. Tiež ju môžeme presunúť za ostatné lentilky pomocou príkazu späť o niekoľko vrstiev.

Nástroje na interaktívnu prácu s postavami

Príkazy pre tmavosivú lentilku

Čo sme sa naučili

V prostredí *Scratch* sme sa naučili pracovať v priamom režime, vytvárať svoje vlastné scenáre pre jednu alebo aj niekoľko postáv. Vieme tiež testovať niektoré udalosti a riadiť postavy pomocou klávesov. Dokážeme testovať kliknutia myši a vieme pracovať s premennými. V rámci aktivít sme pripravovali jednoduché hry, ktoré môžeme využiť priamo na hodinách informatiky, ktoré sú venované základom programovania na 2. stupni ZŠ.

6 Prínos malých jazykov

Zoznámili sme sa s piatimi malými programovacími jazykmi vhodnými na výučbu programovania. Okrem týchto jazykov poznáte z predchádzajúcich modulov programovací jazyk Imagine Logo a programovací jazyk NXT-G, ktorý sa používa na programovanie robotických stavebníc Lego MindStorms Education.

Využite svoje poznatky a skúsenosti v diskusiách a zhodnoťte prínos malých programovacích jazykov.

Diskusia 1	Diskutujte o vlastnostiach malých jazykov vhodných na vyučovanie programovania: • personifikácia procesora a vizualizácia výpočtu (aké postavy vykonávajú programy, ako sa zobrazuje priebeh výpočtu), • zameranie jazyka – malé jazyky nie sú univerzálne, sústreďujú sa na nejaký jav, techniku programovania, oblasť využitia, • jednoduchosť jazyka, nízky prah osvojenia, • spôsob zápisu programu (textový, ikonický, skladačkový), • dostupnosť, podpora, zdieľanie projektov na internete.
Diskusia 2	Aké sú vaše skúsenosti s využívaním malých programovacích jazykov vo vyučovaní programovania? Ktoré jazyky by mohli byť užitočné a dostatočne motivujúce na základnej, na strednej škole? Aké iné malé jazyky poznáte resp. používate na vyučovaní programovania?

Čo sme sa naučili v tomto module

Zhrnutie

Poznáme niekoľko programovacích jazykov vhodných na vyučovanie základov programovania:

Karel je jednoduchý textovo orientovaný programovací jazyk, ktorý je zameraný na základy štruktúrovaného programovania.

Baltík je ikonický programovací jazyk, v ktorom sa dajú vytvárať graficky príťažlivé, aj interaktívne projekty s využitím multimédií.

IzyLogo je jednoduchý programovací jazyk typu Logo s korytnačou grafikou. Skladačkový spôsob programovania vylučuje syntaktické chyby. Prostredie umožňuje pripravovať zadania pre žiakov.

Živý obraz je prostredie na propedeutiku (úvod do) objektového a udalosťami riadeného programovania. V programoch môžu bežať paralelné procesy. Spôsob zápisu programu je ikonický.

Scratch je programovací jazyk so skladačkovým zápisom programov. Programujú sa reakcie objektov (postáv) na rôzne udalosti. Vytvorené projekty je možné publikovať na webe a spúšťať aj vo webovom prehliadači.

Literatúra a použité zdroje

[1] Blaho, A. (1989) *Detské programovacie jazyky III. Karel-3D*. Metodický list k práci záujmových útvarov programovania. Bratislava: Smena, 1989.

[2] Gašparovičová, Ľ. – Hvorecký, J. (1991) *Kamaráti robota Karla*. Bratislava: Mladé letá, 1991. ISBN 80-06-00421-8

[3] Brusilovsky, P. – Calabrese, E. – Hvorecký, J. – Kouchnirenko, A.- Miller, P. (1997) Minilanguages: A Way to Learn Programming Principles. In: *Education and Information Technologies*. 1997, vol. 2, no. 1, p. 65-83

[4] Grédi, L. (2004). *Karel for Win32*. Diplomová práca. Nitra: Fakulta prírodných vied Univerzity Konštantína Filozofa v Nitre, 2004.

[5] Pecinovský, R. – Vácha, J.: Baltík. Učebnica programovania nielen pre deti. SGP Systems.

[6] SGP Systems Výukové programovací nástroje C# pro děti, mládež i dospělé. [online] Dostupné na internete: http://www.sgpsys.com/cz/

[7] TIB Tvorivá informatika s Baltíkom. [online] Dostupné na internete: http://www.tib.sk/

[8] Projekt Tvořivý učitel tvořivé informatiky realizovaný v TIB občanské sdružení v Prahe, časť Metodika. [online] Dostupné na internete: http://tib.cz/tvorivyucitel/metodika.htm

[9] Tomcsányiová, M. a kol.: Riešenie problémov a základy programovania 1, 32 s., Zvolen, Bratia Sabovci, 2009, ISBN 978-80-8118-023-1.

[10] Tomcsányiová, M. a kol.: Riešenie problémov a základy programovania 2, 28 s., Zvolen, Bratia Sabovci, 2010, ISBN 978–80–8118–029-3.

[11] Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., Kafai, Y. (2009). Scratch: Programming for All. Communications of the ACM, November 2009.

[12] Scratch. Imagine – program – share. http://scratch.mit.edu/

[13] Scratch Resources. http://resources.scratchr.org/

[14] ScratchEd. Learn – share – connect. http://scratched.media.mit.edu/

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

Autori © RNDr. Gabriela Lovászová, PhD.
Ing. Ľudmila Galbavá
PaedDr. Viera Palmárová, PhD.
PaedDr. Monika Tomcsányiová, PhD.

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

Podnázov Malé programovacie jazyky

Študijný materiál prešiel recenzným pokračovaním.

Recenzenti PaedDr. Miloslava Sudolská, PhD.
PaedDr. Daniela Bezáková, PhD.

Počet strán 36

Náklad 300 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 978-80-8118-066-8