Ukážky implementácie nástrojov pre formatívne hodnotenie vo vyučovaní informatiky

Teoretické východiská

Informatické myslenie

Súčasné vyučovanie informatiky sa zameriava nielen na rozvíjanie informatických vedomostí a zručností, ale aj na rozvíjanie nadpredmetových kompetencií – konceptov **informatického myslenia**. (angl. computational thinking). Pri realizácii výučby sa využívajú vyučovacie prístupy (napr. bádateľsky orientovaná výučba, projektové vyučovanie, problémové vyučovanie) zamerané na aktívne zapojenie sa žiaka do procesov jeho vzdelávania sa a reflexie výsledkov vzdelávania sa. Nevyhnutnou súčasťou takto orientovanej výučby sú nástrojov pre formatívne hodnotenie, ktoré umožňujú žiakovi efektívne uvedomovať si, riadiť a plánovať proces vlastného vzdelávania sa.

Informatické myslenie sa často poníma ako súbor na predmete nezávislých schopnosti žiakov nevyhnutných pre vzdelávanie a život v 21. storočí. Informatické myslením môžeme podľa (BCS Barefoot, 2019) vymedziť ako efektívne riešenie problémov s pomocou počítača alebo bez neho. Informatické myslenie je komplexná schopnosť charakterizovaná viacerými **konceptmi**. Pre implementáciu informatického myslenia do vyučovania informatiky sme vybrali rámec navrhnutý Computing at School (BCS, 2021) pozostávajúci zo šiestich konceptov:

- **Logika** (predpovedať, analyzovať)
  - logicky vyvodzovať závery z pozorovaní a experimentov,
  - analyzovať systémy alebo programy a predpovedať ich správanie sa.
- **Algoritmy** (vytvárať kroky a pravidlá)
  - zostavovať postupy činností pre vykonávateľa (algoritmy, programy, scenáre, storyboardy),
  - využívať, upravovať, vylepšovať vytvorené postupy (algoritmy, programy, scenáre, storyboardy).
- **Dekompozícia** (rozdeľovať na časti)
  - rozdeľovať problémy na jednoduchšie podproblémy, riešenie ktorých bude využiteľné pri riešení pôvodných problémov.
- **Hľadanie vzorov** (rozpoznávať a využívať podobnosti)
  - rozpoznávať a určovať v systémoch rovnaké či podobné časti, vlastnosti alebo pravidlá,
  - využívať nájdené vzory pri rôznych činnostiach alebo riešení problémov.
- **Abstrakcia** (vyberať podstatné)
  - určovať, ktoré detaily, prvky, vlastnosti alebo vzťahy systémov sú v danej situácii podstatné a ktoré môžeme zanedbať.
- **Vyhodnotenie** (robiť rozhodnutia)
  - určovať relevantné kritériá hodnotenia,
  - vyhodnocovať projekty, programy alebo algoritmy podľa vybraných kritérií.

Vybrané metódy vyučovania

Pri implementácii nástrojov pre formatívne hodnotenie do výučby treba zvažovať aj ich použitie v jednotlivých fázach vyučovania. Pri tradičnom vyučovaní informatiky uvažujeme štyri fázy – **motivačnú**, **expozičnú**, **fixačnú** a **diagnostickú**.

Bádateľsky orientované vyučovanie – učebný cyklus 5E

V bádateľsky orientovanej výučbe informatiky využívame učebný cyklus 5E s piatimi fázami:

- **Zapojenie** (angl. Engage):

  Cieľom tejto fázy je motivovať žiakov pre skúmanie uvedenej oblasti a zistiť ich prvotné predstavy o skúmanej problematike. Veľmi často ide o frontálnu formu výučby vedenú učiteľom napr. uvedenie krátkeho príbehu, zadanie zaujímavej úlohy, diskusiu. Aktivity v tejto fáze by mali smerovať k tomu, aby žiaci dospeli k tomu, čo je pre nich zaujímavé alebo potrebné preskúmať, a získali vlastnú predstavu o cieľoch vyučovacej hodiny. Prvotné predstavy žiakov o skúmanej problematike vieme zistiť pomocou riešenia úloh a diskusie so žiakmi.

- **Skúmanie** (Explore):

  V tejto fáze žiaci realizujú viaceré podnetné aktivity, pomocou ktorých prichádzajú k rôznym predstavám a hypotézam ako funguje skúmaný systém. Veľmi dôležitá je podpora od učiteľa prostredníctvom rôznych pomôcok napr. kódov programov, diagramov, tabuliek, návodov a nástrojov pre formatívne hodnotenie napr. kladením otázok, vypĺňaním predikčných kariet. hlavne učiteľovým slovným usmernením či provokatívnymi otázkami, nie učiteľovým predkladaním hotových riešení. Učiteľ v tejto fáze prechádza pomedzi žiakov a sleduje ich prácu – v prípade nejasností im objasní zadanie úlohy, preformuluje úlohu alebo im pomôže doplňujúcimi otázkami. Úlohy v tejto fáze sú často formulované spôsobom „preskúmajte čo robí program/ako funguje systém“, „doplňte program, aby robil ...“, pri ktorých žiaci majú k dispozícii pracovné súbory – programy či údaje. Odpovede žiakov v pracovných listoch zachycujú aktuálny stav ich poznania, majú možnosť uviesť, že daný prvok učiva nevedia, čo súvisí s rozvíjaním ich metakognície. V každej metodike uvádzame kľúčové prvky, na ktoré je skúmanie zamerané. V tejto fáze nemusia žiaci nutne objaviť a preskúmať všetko s čím sa v metodike stretnú. Napr. stačí ak objavia, že reťazce majú metódy (na príklade dvoch metód). V tradičnej výučbe sa táto fáza vynecháva.

- **Vysvetlenie** (Explain):

  V tejto fáze by si mali žiaci „poupratovať“ vo vlastnej hlave, uvedomiť si k čomu dospeli a formulovať svoje predstavy o tom ako funguje skúmaný systém, ktorý skúmali v predchádzajúcej fáze. Učiteľ vyzýva všetkých žiakov, aby vysvetlili ako rozumejú práve osvojovanému učivu. Diskutuje so žiakmi a koriguje ich predstavy o skúmanej problematike. Na konci žiackeho vysvetľovania zhrnie zažité skúsenosti žiakov a pomenuje ich odborným slovníkom. Vo fáze Vysvetlenia by sme mali sledovať len zopár hlavných cieľov a nezaťažovať žiakov nepodstatnými detailmi. Preto v metodikách zámerne neuvádzame úplný zoznam príkazov a ich variácií. Niektoré príkazy prezradíme žiakom až keď nastane vhodná situácia, kedy sú potrebné (napr. `hideturtle()` či `delay(0)`). V tejto časti učiteľ môže použiť rôzne pomôcky (tabuľky, diagramy, grafy, zdrojové programové kódy) podporujúce pochopenia učiva rôznymi učebnými štýlmi. Túto fázu môže učiteľ doplniť o ďalšie informácie, ktoré môžu žiaci

využiť vo fáze Rozpracovanie. Napríklad okrem metód reťazcov, ktoré sme objavili vo fáze Skúmanie, sú užitočné aj nasledujúce dve-tri metódy (pozor, v žiadnom prípade nenasleduje komplet manuál k reťazcom, možno len odkaz kde sa dá dozvedieť viac). V metodike uvádzame, o čo by mal učiteľ vysvetlenie žiakov doplniť. Z tejto časti by si mali žiaci spraviť výťah a ten zaznamenať do pracovných listov.

- **Rozpracovanie** (Elaborate):

  Podobne ako Skúmanie aj táto fáza je viac časovo dotovaná s viacerými úlohami pre žiakov. Cieľom tejto fázy je, aby si žiaci precvičili a prehĺbili osvojované učivo. Okrem analytických úloh typu „analyzujte program“, „nájdite a opravte chybu v programe“ predkladáme žiakom aj kreatívne úlohy typu „dopracujte/rozšírte program“, „vytvorte program“. Riešenia týchto úloh sú pre učiteľa zdrojom pre získanie predstavy o žiackych miskoncepciách. Aj v tejto fáze je možná vzájomná spolupráca žiakov.

- **Vyhodnotenie** (Evaluate):

  Aj keď počas hodiny učiteľ používa viaceré formy formatívneho hodnotenia (napr. spätnú väzba k riešeniu úloh, diskusiu k žiackym vysvetleniam učiva), na záver hodiny by mal učiteľ poskytnúť žiakom príležitosť k vyjadreniu subjektívneho sebaponímania úrovne osvojených poznatkov a získaných skúseností (napr. sebahodnotiacou kartou) či získaniu objektívnej spätnej väzby k úrovni osvojených poznatkov a získaných skúseností (napr. riešením úlohy didaktického testu). Je veľmi dôležité, aby učiteľ poskytol žiakom čo najskoršiu spätnú väzbu k ich úrovni osvojenia si nových poznatkov.

Projektové vyučovanie

Podľa (Čapek, 2015) **projekt** predstavuje sofistikovanú úlohu zameranú na praktický produkt s jedinečným riešením, ktoré vyžaduje od žiaka autorský vklad. Charakteristickým znakom projektového vyučovania je **prebratie zodpovednosti** za riešenie projektu **samotnými žiakmi**, ktorí majú možnosť samostatne rozhodnúť, ako celý projekt realizovať, čoho dôsledkom sú vzájomne **odlišné produkty** žiakov. Ďalším typickým znakom projektu je výrazná **medzipredmetovosť** (projekt využíva poznatky a zručnosti z rôznych predmetov, tematických oblastí alebo odborov) a taktiež **prepojenosť s realitou** (žiaci riešia konkrétny problém alebo produkt zo života, nie rutinnú nezáživnú úlohu nepoužiteľnú v praxi). Projektové vyučovanie tak vytvára priestor pre efektívnu integráciu poznatkov a prepojenie na rozvoj kľúčovým kompetencií žiaka. Umožňuje rozvíjať kreativitu a samostatnosť žiakov. Súčasťou projektu je potreba samostatného objavovania poznatkov žiakmi počas riešenia. Na základe uvedeného vyplýva, že projektové vyučovanie:

- nie je žiacky referát, pri ktorom žiaci vyhľadávajú na zadanú tému informácie na internete a spracúvajú ich do výslednej prezentácie alebo plagátu (nie je to reálny produkt zo života),
- nie je zadanie samostatnej alebo skupinovej aplikačnej úlohy po predošlom výklade učiteľa (žiaci nemusia objavovať nové poznatky),
- nie je zadanie úlohy, pri ktorej žiaci nemajú možnosť individuálnej voľby činnosti v rámci riešenia (výstup očakáva, že všetci žiaci budú kresliť, písať, programovať a pod.)

- nie je postupnosť krátkych učiteľom zadaných úloh, aj keď ich výsledkom môže byť v praxi užitočný produkt (žiak nedospeje k produktu na základne vlastného plánovania a rozhodovania).

Riešenie projektu pozostáva zo štyroch etáp (Turek, 2008):

- **Voľba témy projektu** – jej špecifikácia, určenie cieľov – produktov projektu,
- **Plánovanie riešenia projektu** – vypracovanie postupu, určenie čiastkových úloh, rozdelenie úloh medzi riešiteľov projektu, určenie termínov jednotlivých úloh projektu,
- **Riešenie projektu** – realizácia plánu projektu žiakmi, učiteľ je v roli pomocníka, oponenta, podnecovateľa atď.
- **Zverejnenie výsledkov riešenia projektu** – prezentácia výsledkov a zhodnotenie práce na projekte.

Ukážky implementácie jednotlivých nástrojov pre formatívne hodnotenie vo vyučovaní informatiky

Ďalej v texte uvádzame ukážky implementácie nástrojov pre formatívne hodnotenie v konkrétnych témach školskej informatiky:

- **Programovanie v Pythone – korytnačia grafika**. Použitie nástrojov pre formatívne hodnotenie (kladenie otázok, predikčná karta, sebahodnotiací test) uvádzame v kontexte celej vyučovacej hodiny.
- **Programovanie v Pythone – komplexný projekt**. Použitie nástrojov pre formatívne hodnotenie (kladenie otázok, rubrika, sebahodnotiaca karta) uvádzame v kontexte štyroch po sebe nasledujúcich vyučovacích hodín.
- **Programovanie v App Inventore – hra guľka**. Použitie nástrojov pre formatívne hodnotenie (sebahodnotiaca karta) uvádzame v kontexte celej vyučovacej hodiny.
- **Kompresia obrázkov**. Použitie nástrojov pre formatívne hodnotenie (predikčná karta, karta k tvorbe záverov) uvádzame v kontexte časti vyučovacej hodiny.
- **Špecifiká aritmetiky počítača**. Použitie nástrojov pre formatívne hodnotenie (Predtým a potom, predikčná karta, karta k tvorbe záverov) uvádzame v kontexte časti vyučovacej hodiny.

Programovanie v Pythone – korytnačia grafika

Korytnačia grafika je témou 2. hodiny z 27 hodinového základného kurzu programovania na strednej škole. Predpokladáme, že z predchádzajúcej hodiny žiaci vedia zadávať príkazy jazyka Python v konzole vybraného vývojového prostredia. Žiaci majú k dispozícii pracovný list a pracovné súbory.

Žiaci by si mali na tejto hodine osvojiť nasledovné vedomosti a zručnosti:

- vytvoriť Python projekt,
- vytvoriť a spustiť jednoduchý Python program a uložiť ho do priestoru projektu ako súbor s príponou py,
- vytvoriť program vykresľujúci obrázok tvorený čiarami a kruhmi rôznej farby a rozmerov použitím základných príkazov korytnačej grafiky (napr. `forward()`, `left()`, `penup()`, `pensize()`, `dot()`, `pencolor()`, `bgcolor()`).

Žiaci by si mali rozvíjať nasledovné koncepty informatického myslenia:

- logika
  - využitím logických zdôvodnení predpovedať správanie sa jednoduchých programov (úlohy 1, 5),
  - využitím logických zdôvodnení detegovať a opravovať chyby v programoch a algoritmoch (úlohy 2),
- algoritmy
  - vytvárať vlastné algoritmy riešiace problém/časti problému (úlohy 3, 4, 6).

Výučba sa realizuje v nasledovných 5 fázach (podľa modelu 5E):

- Zapojenie (cca 4 minúty) – diskusia k významu tvorby obrázkov pomocou programovania.
- Skúmanie (cca 15 minút) – skúmanie príkazov korytnačej grafiky a vývojového prostredia (úlohy 1 až 3).
- Vysvetlenie (cca 5 minút) – diskusia a zhrnutie významu grafických príkazov a základných možností vývojového prostredia.

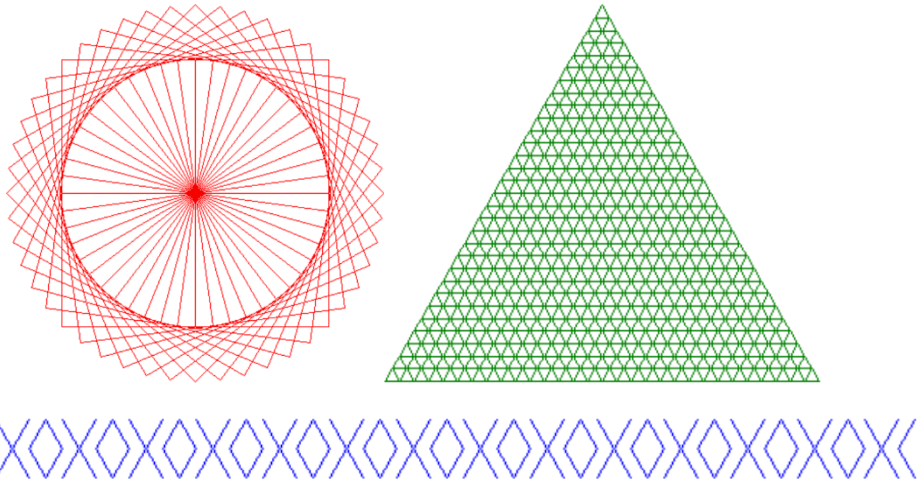
- Rozpracovanie (cca 11 minút) – programovanie grafických problémov (úlohy 4 až 6).
- Vyhodnotenie (cca 5 minút) – vyriešenie sebahodnotiaceho testu so spätnou väzbou od učiteľa.

Zapojenie

Začneme diskusiou s celou triedou (použijeme nástroj pre formatívne hodnotenie **Kladenie otázok**), ktorou smerujeme žiakov k poznaniu, že obrázky sa dajú kresliť nielen rukou, či v grafických editoroch, ale aj programovať. Ide hlavne o tie obrázky, ktoré vykazujú určitú mieru pravidelnosti a zložitosti štruktúry (veľakrát sa opakujúce vzory, či vzory vnorené do iných vzorov). Týmto chceme dosiahnuť motiváciu žiakov, aby sa naučili používať programovací jazyk aj na kreslenie obrázkov.

Možné otázky učiteľa v diskusii:

1. Pomocou ktorých nástrojov viete kresliť obrázky?
   (Možné odpovede žiakov: ceruzka, pero, štetec, prst, grafický editor, programovací jazyk)
2. Pomocou akého nástroja by ste vykreslili uvedené tri obrázky?
   (Možné odpovede žiakov: vektorový editor, programovací jazyk)



3. Závisí výber kresliaceho nástroja od podoby obrázka?
   (Možné odpovede žiakov: áno – ak je zložitý a pravidelný obrázok použijem počítač a pri nepravidelnom kreslím voľnou rukou)
4. Ktoré obrázky je lepšie naprogramovať ako prácne kresliť v grafickom editore?
   A naopak, ktoré obrázky je lepšie nakresliť v grafickom editore ako naprogramovať?

Skúmanie

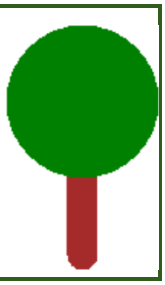
V tejto časti žiaci pracujú samostatne bez pomoci učiteľa. Postupne predikujú a skúmajú (použijeme nástroj pre formatívne hodnotenie **Tvorba predpovedí**) správanie sa vybraných príkazov korytnačej grafiky (úloha 1). Oboznámia sa s vývojovým prostredím, v ktorom otvoria hotový program, ktorý preštudujú, spustia ho a upravia, aby vykreslil požadovaný obrázok (úloha 2). Napokon samostatne vytvoria vlastný projekt a vlastný program na vykreslenie jednoduchého obrázku (úloha 3). Dôležité je, aby v tejto časti hodiny mali žiaci čo najviac príležitosti na priame a aktívne poznávanie s minimálnou a skôr našou individuálnou pomocou (bez konkrétnych odborných rád, skôr povzbudenie či všeobecne usmernenie), aby mali radosť z vlastného skúmania a objavovania.

Úloha 1 *Preskúmajte, čo sa stane, ak do konzoly postupne zadáte príkazy jazyka Python uvedené v tabuľke. Vedľa príkazu napíšte svoju predpoveď aj zistenú skutočnosť. (Zadávajte každý príkaz do konzoly samostatne (aj keď sú v jednej bunke tabuľky 2 riadky) a na potvrdenie príkazov stlačte kláves ENTER).*

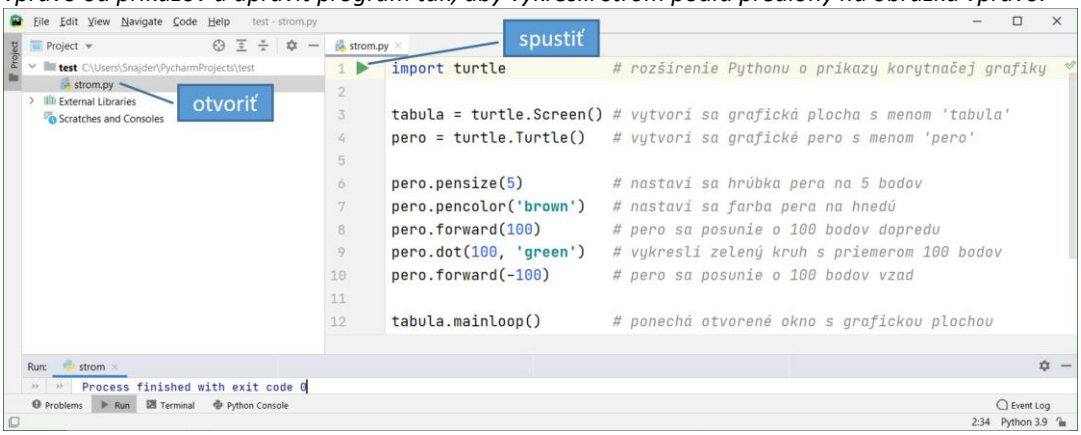
#	Príkaz(y)	Aký výsledok predpovedáte?	Čo ste zistili po spustení príkazov?
1.	`forward(100)`		Neurobilo nič, len vypísalo chybovú hlášku: NameError: name 'forward' is not defined
2.	`import turtle` `pero = turtle.Turtle()`		Spustilo sa okno "Python Turtle Graphics" s grafickým perom natočeným na východ
3.	`pero.forward(100)`		Grafické pero vykreslilo úsečku dĺžky 100 bodov
4.	`pero.left(60)`		Grafické pero sa otočilo vľavo o 60 stupňov
5.	`pero.penup()` `pero.forward(100)`		Grafické pero sa zdvihlo (vyplo) a posunulo bez kreslenia o 100 bodov dopredu
6.	`pero.pendown()` `pero.backward(100)`		Grafické pero sa priložilo na plochu (zaplo) a posunulo (s kreslením) o 100 bodov vzad
7.	`pero.clear()`		Z kresliaceho plátna sa zmazalo všetko čo nakreslilo grafické pero

Žiaci riešia **úlohu 1**, v ktorej najprv predpovedajú výsledok spustenia uvedených príkazov. Následne experimentovaním – spustením uvedených príkazov v konzole overujú svoje predpovede. Príkaz `forward()` uvedený v 1. riadku tabuľky nie je štandardnou súčasťou jazyka Python, preto interpreter zahlási chybovú správu ”NameError: name 'forward' is not defined”. Príkazy v 2. riadku umožnia využívať grafické príkazy uložené v module **turtle**. Po úspešnom odoslaní príkazu `pero = turtle.Turtle()` si žiaci štandardne nevšimnú spustenie nového okna s grafikou, na čo ich treba upozorniť. V nasledujúcich riadkoch tabuľky sú uvedené ďalšie základné príkazy korytnačej grafiky, ktorých význam žiaci preskúmajú: `forward()`, `backward()`, `left()`, `penup()` a `clear()`.

Úloha 2 *Janko chce v editore kódu naprogramovať strom uvedený na obrázku. Jeho program **strom.py** však nevykresľuje strom podľa predlohy. Pomôžte mu upraviť jeho program, aby správne vykreslil uvedený strom.*
*Odporúčame, aby ste najprv otvorili program **strom.py** a spustili jeho kód kliknutím na uvedené oblasti v dolnom obrázku. Program vieme spustiť aj stlačením klávesov SHIFT + F10. Následne by ste mali preskúmať vysvetľujúce jednoriadkové komentáre*



vpravo od príkazov a upraviť program tak, aby vykreslil strom podľa predlohy na obrázku vpravo.



Riešenie:

Pôvodné riešenie upravíme na správne, ak pred riadok 6 doplníme príkaz `pero.left(90)` *lebo vykreslený je zle orientovaný, v riadku 6 zmeníme parameter na 20:* `pero.pensize(20)` *lebo hrúbka pera je príliš malá, a pred riadok 9 doplníme príkaz* `pero.penup()`*, lebo pri návrate do pôvodnej pozície sa koruna stromu prekreslila kmeňom. Výsledný kód bude vyzerať nasledovne s vyznačením dvomi znakmi # v komentároch tých riadkov, v ktorých boli urobené uvedené úpravy:*

```
import turtle              # rozšírenie Pythonu o príkazy korytnačej grafiky

tabula = turtle.Screen()   # vytvorí sa grafická plocha s menom 'tabula'
pero = turtle.Turtle()     # vytvorí sa grafické pero s menom 'pero'

pero.left(90)              ## natočenie pera vľavo o 90 stupňov
pero.pensize(20)           ## nastaví sa hrúbka pera na 20 bodov
pero.pencolor('brown')     # nastaví sa farba pera na hnedú
pero.forward(100)          # pero sa posunie o 100 bodov dopredu
pero.penup()               ## zdvihnutie grafického pera
pero.dot(100, 'green')     # vykreslí zelený kruh s priemerom 100 bodov
pero.forward(-100)         # pero sa posunie o 100 bodov vzad

tabula.mainloop()          # ponechá otvorené okno s grafickou plochou
```

Úloha 2 je zameraná, jednak na samostatné získanie prvých skúseností s otvorením a spustením hotového programového kódu (uloženého v pracovnom súbore **strom.py**), a jednak na preskúmanie programového kódu a jeho úpravu, aby sa vykreslil strom podľa predlohy. Pri riešení úlohy žiaci objavia význam nových grafických príkazov `pensize()`, `pencolor()`, `dot()`. Mali by tiež pochopiť rovnocennosť príkazov `forward(-100)` a `backward(100)`. Oboznámia sa so syntaxou jednoriadkových komentárov, ktoré umožňujú autorom zdokumentovať význam častí zapísaného programového kódu. Získajú prvotnú predstavu o štruktúre grafického programu, jeho úvodnej inicializačnej časti, strednej výkonnej časti a záverečnej časti.

Úloha 3 *Podľa poskytnutého návodu vytvorte vo vývojovom prostredí projekt **kreslenie** a program na vykreslenie veľkého tlačeného písmena L. Vytvorený program uložte v Python súbore s menom **elko.py**.*

a) Najprv pouvažujte a uveďte, ktoré grafické príkazy použijete na vykreslenie písmena L:
*b) Po vytvorení programu, uveďte adresu, kde ste uložili súbor **elko.py** na vašom lokálnom disku:*

Riešenie:

a) `forward()`, prípadne `backward()` a `left()`
b) cesta k súboru elko.py

```
import turtle              # rozšírenie Pythonu o príkazy korytnačej grafiky

tabula = turtle.Screen()   # vytvorí sa grafická plocha s menom 'tabula'
pero = turtle.Turtle()     # vytvorí sa grafické pero s menom 'pero'

pero.forward(50)           # vykreslí krátke vodorovné rameno písmena L
pero.forward(-50)
pero.left(90)              # natočí sa pero na sever
pero.forward(100)          # vykreslí dlhé zvislé rameno písmena L
pero.forward(-100)

tabula.mainloop()          # ponechá otvorené okno s grafickou plochou
```

Žiakom poskytneme návod a necháme ich vyriešiť úlohu 3 zameranú na vytvorenie projektu **kreslenie** a programu **elko.py** vykresľujúceho veľké písmeno L. Pri riešení úlohy musia žiaci zostaviť postupnosť príkazov na vykreslenie písmena L a na konci spustiť vytvorený program. Rovnako majú vedieť určiť adresu vytvoreného projektu a programu na vlastnom počítači. Tu by si mali žiaci uvedomiť, že všetky programy sú súčasťou projektu.

Vysvetlenie

V ďalšej časti hodiny necháme najprv žiakov, aby nahlas vysvetlili tri nimi skúmané záležitostí:

- význam jednotlivých príkazov korytnačej grafiky (uvedených v tabuľke úlohy 1),
- štruktúru programu na vykreslenie obrázkov pomocou korytnačej grafiky (uvedenú v úlohe 2),
- postupnosť krokov pri vytváraní programov na vykreslenie obrázkov (úloha 3).

Najprv žiakom uvedieme, že vykresľovanie obrázkov postupným zapisovaním príkazov v konzole je veľmi nepraktické, obzvlášť keď sa pomýlime a musíme odznova zapisovať grafické príkazy. Vhodnejšie je použiť editor kódu a postupnosť príkazov uložiť do súboru (s príponou py).

Následne zhrnieme, že príkazy korytnačej grafiky nie sú základnou súčasťou jazyka Python, ale sú dostupné importovaním z knižnice príkazov (modulu) **turtle**. Pri kreslení používame príkazy pre grafické pero (`forward()`, `left()`, ...), ktorým kreslíme na grafickú plochu. Ak chceme nechať otvorené okno s kresbou, použijeme príkaz `mainloop()` pre grafickú plochu.

Pred samotným zapisovaním programu do editora kódu je dôležité, aby sme najprv načrtli a analyzovali obrázok s vyznačením uhlov a dĺžok čiar. Pri kreslení útvarov napr. n-uholníkov nepoužívame vnútorné, ale vedľajšie (susedné) uhly, ktoré sú ich doplnkom do 180 stupňov.

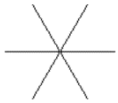
Pri práci vo vývojovom prostredí je dôležité vedieť vytvoriť projekt a súbor v projekte, spustiť a zastaviť beh programu, poznať miesto uloženia projektu a programu, uvedomiť si, že grafický výstup sa realizuje v samostatnom okne.

Pri vytváraní programu je dôležité pomenovať program výstižným názvom, aby sme mali prehľad o vytvorených programoch v danom projekte.

Rozpracovanie

V tejto časti hodiny si žiaci prostredníctvom analytickej úlohy (**úloha 5**) a dvoch úloh na tvorbu programového kódu (**úlohy 4** a **6**) z pracovného listu, precvičia použitie základných príkazov korytnačej grafiky a niektorých stratégií riešenia problémov (napr. nakresli si obrázok, vyskúšaj a over, rozdeľ problém do podproblémov). Pri riešení úlohy 5 žiaci predikujú výsledok uvedeného programu, pričom svoju predikciu spustením programu overia (použijeme nástroj pre formatívne hodnotenie **Tvorba predpovedí**, pričom sme nevyužili formálnu podobu predikčnej karty).

Úloha 4 *Vytvorte program* ***vlocka.py*** *na vykreslenie vločky uvedenej na obrázku:*



Riešenie 1

```
# rameno 50
pero.forward(50)
pero.forward(-50)
pero.left(60)

# rameno 50
pero.forward(50)
pero.forward(-50)
pero.left(60)

# rameno 50
pero.forward(50)
pero.forward(-50)
pero.left(60)

# rameno 50
pero.forward(50)
pero.forward(-50)
pero.left(60)

# rameno 50
pero.forward(50)
pero.forward(-50)
pero.left(60)

# rameno 50
pero.forward(50)
pero.forward(-50)
pero.left(60)
```

Riešenie 2

```
# rameno 100
pero.forward(50)
pero.forward(-100)
pero.forward(50)
pero.left(60)

# rameno 100
pero.forward(50)
pero.forward(-100)
pero.forward(50)
pero.left(60)

# rameno 100
pero.forward(50)
pero.forward(-100)
pero.forward(50)
pero.left(60)
```

Riešenie 3

```
# rameno 100
pero.forward(100)
pero.penup()
pero.left(120)
pero.forward(50)
pero.left(120)
pero.pendown()

# rameno 100
pero.forward(100)
pero.penup()
pero.left(120)
pero.forward(50)
pero.left(120)
pero.pendown()

# rameno 100
pero.forward(100)
pero.penup()
pero.left(120)
pero.forward(50)
pero.left(120)
pero.pendown()
```

Aj keď je precvičovacia **úloha 4** pomerne ľahká, dá sa riešiť rôznymi spôsobmi. Je zaujímavé, aj keď nie nevyhnutné, priamo vo výučbe analyzovať (napr. z pohľadu počtu vykreslených čiar) k akým rôznym riešeniam prišli žiaci. V tabuľke sú uvedené ukážky troch riešení: kreslením zo stredu 12, resp. 9 čiar bez

zdvihnutia pera (riešenie 1, resp. 2), či kreslením len 3 čiar so zdvíhaním a pokladaním pera na grafickú plochu (riešenie 3).

Úloha 5 *Najprv slovne alebo graficky uveďte aký obrázok vykreslí dole uvedený program:*
*Potom otvorte program **neznamy.py** a overte, či ste uviedli správny obrázok.*

```
import turtle             # rozšírenie Pythonu o príkazy korytnačej grafiky

tabula = turtle.Screen() # vytvorí sa grafická plocha s menom 'tabula'
pero = turtle.Turtle()   # vytvorí sa grafické pero s menom 'pero'
tabula.bgcolor('lightyellow')

pero.forward(100)
pero.left(45)
pero.forward(-20)
pero.forward(20)
pero.left(-90)
pero.forward(-20)
pero.forward(20)
pero.left(45)
pero.forward(-100)

tabula.mainloop()          # ponechá otvorené okno s grafickou plochou
```

Riešenie: Program vykreslí šípku smerujúcu vpravo na žltom pozadí.

Pri **úlohe 5** poskytneme žiakom pracovný súbor **neznamy.py**. Úloha je zameraná na analyzovanie hotového neznámeho programového kódu (vykreslenie šípky smerujúcej doprava na žltej grafickej ploche so žltým pozadím). Žiaci by mali samostatne pochopiť význam nového príkazu `bgcolor()` na nastavenie farby pozadia grafickej plochy.

Úloha 6

Riešte podľa pokynov učiteľa

*Vytvorte program **hodiny.py** na vykreslenie hodín ukazujúcich čas 10:00 a program **trikolora.py** na vykreslenie trojfarebnej trikolóry podľa predlohy na uvedených obrázkoch:*

Uveďte, ako by ste postupovali v prípade vykreslenia slovenskej trikolóry s bielou farbou na okraji.
(Riešenie: Nastavíme pozadie plátna na inú ako bielu farbu alebo ešte pred vykreslením bieleho kruhu vykreslíme čierny kruh s priemerom o niečo väčším ako priemer bieleho kruhu.)
Riešenie 6A:

```
import turtle             # rozšírenie Pythonu o príkazy korytnačej grafiky

tabula = turtle.Screen() # vytvorí sa grafická plocha s menom 'tabula'
pero = turtle.Turtle()   # vytvorí sa grafické pero s menom 'pero'

pero.dot(100, 'yellow')  # ciferník

pero.left(90)            # veľká (minútová) ručička
pero.forward(50)
pero.forward(-50)

pero.left(60)            # malá tučná (hodinová) ručička
```

```
pero.pensize(3)
pero.forward(30)
pero.forward(-30)

pero.hideturtle()         # skrytie grafického pera
tabula.mainloop()         # ponechá otvorené okno s grafickou plochou
```

Riešenie 6B:

```
import turtle             # rozšírenie Pythonu o príkazy korytnačej grafiky

tabula = turtle.Screen() # vytvorí sa grafická plocha s menom 'tabula'
pero = turtle.Turtle()   # vytvorí sa grafické pero s menom 'pero'

pero.dot(100, 'black')   # čierny kruh s priemerom 100
pero.dot(66, 'red')      # červený kruh s priemerom 66
pero.dot(33, 'yellow')   # žltý kruh s priemerom 33

pero.hideturtle()         # skrytie grafického pera
tabula.mainloop()         # ponechá otvorené okno s grafickou plochou
```

Podľa situácie necháme žiakom vyriešiť na hodine časť alebo celú **úlohu 6**. Je zaujímavé zistiť ako vyriešili žiaci vykreslenie trikolóry s bielou farbou na okraji, ktorá má byť rozpoznateľná o pozadia (napr. zmenou farby pozadia na inú ako bielu farbu či vykreslením čierneho kruhu s priemerom o bod väčším ako následného bieleho kruhu).

Príkaz `hideturtle()` na schovanie grafického pera nepredkladáme žiakom v hotovej podobe, ale predstavíme ho podľa potreby – až keď nastane situácia, že žiakom vo vykreslenom obrázku prekáža grafické pero a chcú vidieť obrázok bez neho.

Pri vytváraní a spúšťaní viacerých programov vo vývojovom prostredí sa v Run paneli v dolnom podokne vytvárajú záložky pre každý spustený program. Pri opätovnom spustení či zastavení patričného programu je potrebné sa prepnúť na záložku s jeho menom.

Týmto je zavŕšená precvičovacia časť hodiny. Pre ďalšie prípadné precvičenie programovania kreslenia obrázkov pomocou korytnačej grafiky či na domácu úlohu poslúžia **úlohy 7** až **11** uvedené na konci pracovného listu.

Vyhodnotenie

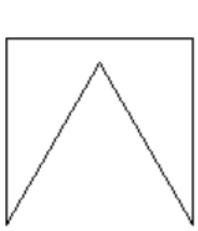
V záverečnej časti hodiny necháme žiakom vyriešiť sebahodnotiaci test (použijeme nástroj pre formatívne hodnotenie **Sebahodnotiaci test**).

Sebahodnotiaci test

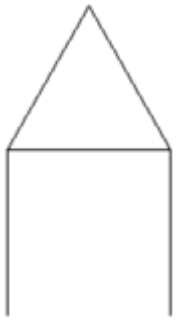
1. Zakrúžkujte, ktorý z obrázkov A, B alebo C sa vykreslí pomocou uvedenej postupnosti príkazov.

```
pero.forward(100)
pero.forward(-100)
pero.right(90)
pero.forward(100)
pero.forward(-100)
pero.left(150)
pero.forward(100)
pero.right(120)
pero.forward(100)
pero.right(30)
pero.forward(100)
```

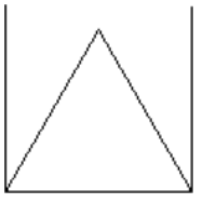
Obrázok A



Obrázok B



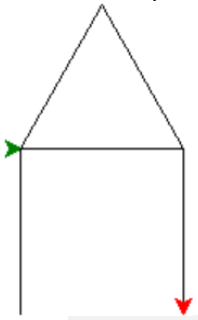
Obrázok C



Vo vybranom obrázku vyznačte štartovaciu a cieľovú pozíciu a natočenie grafického pera.
Uveďte, koľkokrát (.....) bol použitý príkaz `forward()` a koľko (.....) úsečiek je vykreslených na obrázku.
Uveďte o aký celkový uhol sa natočilo grafické pero v cieľovej pozícii oproti štartovacej pozícii:

.....

Riešenie:
Výstupom programu je obrázok B, v ktorom sme vyznačili štartovaciu pozíciu zelenou farbou (pero je natočené na východ) a cieľovú pozíciu červenou farbou (pero je natočené na juh).



Príkaz `forward()` bol použitý 7-krát.
Na obrázku je vykreslených 5 úsečiek, v dvoch prípadoch sa pero vrátilo po tej istej trase.
Grafické pero sa zo štartovacej pozícii do cieľovej pozície natočilo o 90 stupňov vpravo (90 – 150 + 120 + 30, pričom otáčanie doprava berieme so znamienkom plus a doľava so znamienkom mínus).

Programovanie v Pythone – komplexný projekt

Posledné štyri hodiny zo základného kurzu programovania sú realizované formou projektového vyučovania, v rámci ktorého žiaci využijú svoje doterajšie programovacie vedomosti a zručnosti a vytvoria v praxi využiteľný softvérový produkt.

Žiaci by si mali rozvíjať nasledovné koncepty informatického myslenia:

- logika
  - logicky zdôvodniť rozdelenie algoritmu/programu/problému/objektu na menšie časti (návrh a zdôvodnenie funkcionalít programu pri anotácii projektu, zdôvodniť rozdelenie programu na čiastkové podprogramy pri implementácii riešenia problému),
  - logicky zdôvodniť zmenu algoritmu/programu (zdôvodniť zmenu algoritmu/programu na základe doterajších skúseností pri implementácii riešenia problému alebo z dôvodu refaktorizácie riešenia projektu),
- algoritmy
  - vytvárať vlastné algoritmy riešiace problém (návrh a implementácia vlastných algoritmov pri implementácii riešenia problému),
  - využívať existujúce algoritmy v návrhu vlastných algoritmov (adaptácia existujúcich algoritmov pri implementácii riešenia problému),
  - vylepšovať existujúce algoritmy (zlepšenie efektívnosti, rozšírenie algoritmu na väčšiu množinu vstupov, rozšírenie funkcionality) (optimalizovať použité algoritmy pri riešení konkrétneho problému).
- dekompozícia
  - hierarchická dekompozícia – hierarchicky rozdeliť problémy na menšie časti tak, aby sa dali využiť pre dosiahnutie cieľa (zobraziť hierarchiu, aj podproblémy rozdeliť na menšie podproblémy) (špecifikácia funkcionalít programu a ich častí pri anotácii projektu a pri implementácii riešenia problému),
  - hierarchická dekompozícia – hierarchicky rozdeliť prácu pre členov hierarchického tímu ktorí delegujú prácu na podriadených (pridelenie čiastkových úloh jednotlivým členom tímu pri implementácii riešenia problému).
- hľadanie vzorov
  - zovšeobecniť riešenie podobných problémov na celú triedu, zovšeobecniť na základe konkrétnych prípadov (zovšeobecnenie čiastkových námetov v rámci diskusie pri návrhu témy projektu a pri implementácii riešenia problému),
- abstrakcia
  - určiť, ktoré detaily/prvky/vlastnosti/vzťahy objektov/problémov/procesov sú v danej situácii podstatné a ktoré môžeme zanedbať (identifikácia podstatných funkcionalít programu pri návrhu témy projektu a pri implementácii riešenia problému),
- vyhodnotenie
  - posúdiť kvalitu/správnosť/efektívnosť/vhodnosť postupu na základe vybraných/definovaných kritérií (napr. posúdiť efektívnosť algoritmov, správnosť dekompozície, presnosť a úplnosť algoritmu/programu, testovať program) (čiastkové vyhodnotenie správnosti riešenia problému počas priebežného testovania, celkové vyhodnotenie riešenia problému na základe rubriky, sebahodnotiacej karty a záverečnej diskusie pri prezentácii riešenia projektu).

V priebehu štyroch vyučovacích hodín sa budeme postupne venovať jednotlivým etapám tvorby komplexného projektu:

- Zostavenie tímu, výber témy projektu, analýza a návrh riešenia problému, tvorba anotácie projektu (použijeme nástroj pre formatívne hodnotenie **Kladenie otázok** a **Rubrika** (Tabuľka 1)).
- Implementácia riešenia problému (použijeme nástroj pre formatívne hodnotenie **Rubrika**).
- Finalizácia projektu, sebahodnotenie výsledného projektu (použijeme nástroj pre formatívne hodnotenie **Rubrika** a **Sebahodnotiaca karta** (Tabuľka 2)).
- Prezentácia projektu s diskusiou. Hodnotenie projektu učiteľom a spolužiakmi. (žiakom poskytneme spätnú väzbu vyhodnotením nástrojov pre formatívne hodnotenie **Rubrika** a **Sebahodnotiaca karta**).

Žiaci majú na začiatku riešenia projektu k dispozícii šablónu anotácie projektu a rubriku projektu.

Prvú etapu riešenia projektu začneme diskusiou s celou triedou, v rámci ktorej kladieme žiakom otázky typu: „Pre koho bude výsledný program určený?“, „Aký účel bude plniť výsledný program?“, „S ktorými typmi dát bude pracovať výsledný program?“, ... (použijeme nástroj pre formatívne hodnotenie **Kladenie otázok**). Po tejto diskusii žiakom poskytneme rubriku hodnotenia projektu, s ktorou budú ďalej pracovať v priebehu riešenia projektu (použijeme nástroj pre formatívne hodnotenie **Rubrika**). Navrhujeme už na začiatku žiakom zverejniť rubriku, a to vzhľadom na rôznorodosť typov projektov, veľkosť tímov, výkonnostné rozdiely žiakov. Výsledkom prvej etapy je rozdelenie žiakov do riešiteľských tímov, určenie vedúcich tímov, návrh tém a stručné anotácie projektov.

V druhej etape členovia tímov spresnia špecifikáciu problému (určia vstupy, výstupy a vzťahy medzi nimi, rozdelia problém na podproblémy), navrhnú funkcionality jednotlivých podproblémov a návrh grafického používateľského rozhrania, začnú programovať funkcionality jednotlivých podproblémov. Žiaci pracujú samostatne v tímoch, učiteľ len monitoruje a usmerňuje prácu projektových tímov. Odporúčame učiteľom pripomínať žiakom, že každý člen tímu je dôležitý a zodpovedajúci za svoj diel práce. Rovnako by mali žiaci prípadné zmeny špecifikácie problému konzultovať s učiteľom ako zadávateľom zákazky, aby sa o nich nedozvedel až pri záverečnej prezentácii projektu.

V tretej etape žiaci dokončia implementáciu riešenia všetkých podproblémov, ktoré otestujú na pripravených testovacích dátach a ošetria odolnosť programu voči chybám pomocou výnimiek. Za domácu úlohu si žiaci pripravia stručnú prezentáciu k vytvorenému projektu a vyplnia rubriku (použijeme nástroj pre formatívne hodnotenie **Rubrika**).

V štvrtej etape žiaci prezentujú svoj (tímový) komplexný projekt a diskutujú so spolužiakmi a učiteľom pozitíva projektu a odporúčania k jeho vylepšeniu či rozšíreniu. Na konci hodiny žiaci vyplnia sebahodnotiacu kartu (použijeme nástroj pre formatívne hodnotenie **Sebahodnotiaca karta**).

Tabuľka 1 Rubrika pre hodnotenie riešenia projektu, jeho prezentácie a využitia v praxi

Mená autorov projektu:			**Trieda:**	**Dátum:**	
Úroveň **Položka**	**Úroveň 1** **(1 bod)**	**Úroveň 2** **(2 body)**	**Úroveň 3** **(3 body)**	**Úroveň 4** **(4 body)**	**Spolu**
1 Náročnosť riešenia algoritmického problému, pri riešení žiak použil	nanajvýš jednoduchý sekvenčný algoritmus	algoritmus obsahujúci cykly alebo algoritmus obsahujúci vetvenie	algoritmus obsahujúci cykly a vetvenie	algoritmus obsahujúci vzájomné vnorené cykly a vetvenie	
2 Dátové štruktúry použité pri riešení algoritmického problému	žiadne alebo len jednoduché typy int alebo float alebo boolean	štruktúrovaný typ str	štruktúrovaný typ list	kombinácia viacerých štruktúrovaných typov	
3 Ošetrenie chybových stavov	testujú sa nanajvýš požiadavky na vstup	testujú sa operácie s dátami	testujú sa požiadavky na vstup a operácie s dátami		
4 Reakcia na chybové stavy	v prípade problematického stavu nie je používateľ informovaný alebo je informovaný len systémovou hláškou	v prípade problematického stavu je používateľ informovaný relevantnou hláškou o problematickom stave (napr. aj výpisom do konzoly)	v prípade problematického stavu je používateľ informovaný relevantnou hláškou o problematickom stave prostredníctvom grafického rozhrania		
5 Vstup dát	program nepoužíva vstup alebo len vstup z konzoly	vstupné textové okienko/tlačidlo grafickej aplikácie, myš v grafickej ploche	vstupné dáta zo súboru	kombinácia vstupu zo súboru s ďalším interaktívnym vstupom	
6 Výstup dát	program nepoužíva výstup alebo len výstup do konzoly	výstupné textové okienko/ výstup do grafickej plochy	výstupné dáta do súboru	kombinácia výstupu do súboru v kombinácii s ďalším typom výstupu	

7 Dekompozícia problému	žiadna alebo len jedna funkcia riešiaca celý problém	vhodne navrhnutá funkcia s parametrom alebo funkcia s návratovou hodnotou	správne dekomponovaný problém, vhodne navrhnutá funkcie s parametrami a s návratovými hodnotami	správne dekomponovaný problém, jednotlivé časti problému sa riešia v samostatných funkciách, funkcie sa vzájomne volajú a odovzdávajú si výstupné hodnoty	
8 Kód programu	identifikátory sú nejasné, z ich názvu nie je možné dedukovať ich význam	identifikátory (názvy premenných a funkcií) sú popisné, z ich názvu je možné dedukovať význam ich obsahu, resp. použitia	funkcie obsahujú dokumentačné reťaze popisujúce výsledok funkcie, vstupné a výstupné hodnoty	dokumentačné reťazce sú úplné, vstupy a výstupy sú definované podľa štandardu	
9 Používateľské rozhranie	grafické rozhranie nie je k dispozícii alebo je neprehľadné	grafické rozhranie je prehľadné	je zrejmé, ktoré polia sú vstupné a ktoré výstupné, obsah polí je popísaný (napr. pomocou Label)	výstupné polia sú chránené voči používateľským vstupom	
10 Prezentácia projektu žiakom	žiak vie nanajvýš „prerozprávať“ časti kódu	žiak vie popísať logiku algoritmu a identifikovať jednotlivé časti programu (napr. táto funkcia robí to a to)	žiak vie zdôvodniť použitý algoritmus, postup, žiak vie vysvetliť jednotlivé časti programu	žiak vie popísať hraničné (extrémne, krajné) prípady a ako na ne v programe reaguje, príp. v ktorých situáciách program nebude pracovať správne	
11 Využitie projektu v praxi	použiteľný v praxi po veľkých úpravách	použiteľný v praxi po menších úpravách	použiteľný v praxi bez potrebných úprav		
Spolu					

Poznámka: Aj keď dôraz kladieme na využiteľnosť projektu v praxi, ošetrenie chýb počas výpočtu a dôsledné okomentovanie programového kódu, výsledná klasifikácia projektu bude závislá od súčtu bodov dosiahnutých za jednotlivé položky rubriky. Prevod dosiahnutého skóre na známky necháme na samotných učiteľov (napr. hodnotenie výborný pri dosiahnutí hodnoty aspoň 35 bodov). Pri binárnom hodnotení uspel/neuspel by hranicou úspešnosti mohlo byť dosiahnutie určitej hranice, napr. 20 bodov.

Tabuľka 2 Sebahodnotiaca karta na vyhodnotenie priebehu a výsledkov riešenia projektu

Uveďte na koľko percent hodnotíte **celkovú funkčnosť Vášho programu** vzhľadom k anotácii: ..... %

Uveďte **ďalšie funkcionality programu**, o ktoré by sa dal **doplniť** a **vylepšiť** Váš komplexný projekt:

.....

.....

.....

.....

Uveďte, za ktoré záležitosti pri tvorbe Vášho komplexného projektu ste boli Vy **osobne zodpovedný/á**:

.....

Do akej miery ste **spokojný/á** s **tímovou prácou** a **výsledkom projektu**? Vyberte vhodnú odpoveď pre každú položku:

som S/So	veľmi spokojný/á	skôr spokojný/á	neviem	skôr nespokojný/á	veľmi nespokojný/á
svojou prácou na projekte					
prácou ostatných členov tímu					
výsledným programom					

Uveďte, čo ste sa pri tvorbe projektu **nové naučili**:

.....

Aké sú Vaše postoje k **programovaniu**? Vyberte vhodnú odpoveď pre každú položku:

Programovanie je	určite áno	skôr áno	neviem	skôr nie	určite nie
náročné					
zaujímavé					
dôležité					

Okrem tohto komplexného projektu ste naprogramovali niečo užitočné **aj mimo vyučovania** programovania, napr. hru, pomôcku pre nejaký školský predmet? áno – nie

Ak áno, stručne uveďte o **aké softvérové aplikácie** išlo:

.....

.....

Programovanie v App Inventore – hra guľka

Hra guľka je jednou zo série malých projektov, prípravných aktivít pre programovanie väčších projektov vo vývojovom prostredí MIT App Inventor (určené pre vývoj aplikácií pre mobilné zariadenia s operačným systémom Android).

Hlavným cieľom tejto malej aplikácie je, aby žiak bol schopný vytvoriť aplikáciu využívajúcu na pohyb lopty nakláňanie mobilného zariadenia (orientačný senzor), kolíziu s inou loptou (udalosť kolízie s iným objektom), meranie uplynutého času (metódy vrátenia aktuálneho času a rozdielu dvoch časov) a zaznamenanie najlepšieho času hry do mobilného zariadenia (metódy zápisu/čítania hodnoty do/z lokálnej databázy).

Na začiatku žiaci dostanú dve jednoduché aplikácie na skúmanie použitia nových komponentov a ich metód:

- komponent `Clock` a jeho metódy `Now` a `Duration`,
- komponent `TinyDB` a jeho metódy `GetValue` a `StoreValue`.

Následne pracujú s treťou aplikáciou, v ktorej skúmajú nové komponenty a ich udalosti a vlastnosti:

- komponent `OrientationSensor` a jeho udalosť `OrientationChanged` a vlastnosť `Enabled`.
- Komponent `Ball` a jeho udalosť `CollidedWith` a vlastnosti `X`, `Y`, `Visible` a `PaintColor`.

V tejto aplikácii sa guľka pohybuje po obrazovke v závislosti od naklonenia mobilného zariadenia a po kolízii s inou guľka sa skryje. Úlohou žiakov je na základe predchádzajúcich skúmaní doprogramovať tretiu aplikáciu tak, aby mala nasledovné funkcionality:

- nakláňaním mobilného zariadenia sa snažíme dostať malú guľku do väčšej kruhovej jamky,
- ak dostaneme guľku do jamky, guľka sa schová, jamka sa zafarbí na červeno a hra končí,
- po skončení hry sa zaznamená čas trvania hry do lokálnej databázy, ale len vtedy, ak je v hre dosiahnutý čas menší ako čas predtým uložený do databázy.

Po naprogramovaní hry s určitou mierou pomoci učiteľa, žiaci vyplnia sebahodnotiace karty (použijeme nástroj pre formatívne hodnotenie **Sebahodnotiaca karta**), v ktorých sledujeme deklaratívne (Tabuľka 3) a procedurálne vedomosti (Tabuľka 4).

Tabuľka 3 Sebahodnotiaca karta zameraná na deklaratívne vedomosti pri programovaní v App Inventore

Viem:	zatiaľ neviem	viem s pomocou	viem
Vysvetliť vzťah medzi hodnotami komponentu `OrientationSensor` a naklonením mobilného zariadenia			
Vysvetliť kedy nastane udalosť `Ball.CollidedWith`			
Vysvetliť význam vlastnosti `Ball.X` a `Ball.Y`			

Vysvetliť rozdiel medzi vlastnosťami `Ball.Enable` a `Ball.Visible`			
Vysvetliť význam metódy `Clock.Duration`			
Vysvetliť význam dvojice **kľúč**:**hodnota** pri práci s komponentom `TinyDB`			

Tabuľka 4 Sebahodnotiaca karta zameraná na procedurálne vedomosti pri programovaní v App Inventore

Viem vytvoriť hru, v ktorej:	**s veľkou pomocou**	**s malou pomocou**	**samostatne**
Pohyb lopty sa určuje nakláňaním mobilného zariadenia			
Registrujem zrážku dvoch lôpt			
Meriam trvanie hry			
Uchovávam trvanie najkratšej hry aj po ukončení hry			

Pri vypĺňaní sebahodnotiacich kariet by si mali žiaci uvedomiť a zafixovať nové učivo. Sebahodnotiacu kartu môže učiteľ použiť na stručnú sumarizáciu nového učiva a na základe analýzy žiackych odpovedí prispôsobiť svoju nasledovnú pedagogickú intervenciu.

Kompresia obrázkov

Žiaci majú základnú predstavu o princípoch komprimácie súborov a o rôznych reprezentáciách grafickej informácie v závislosti od spôsobu jej kódovania (typu súboru). Poznatky žiakov z týchto dvoch oblastí však môžu byť vzájomne izolované. Žiaci si neuvedomujú, že niektoré grafické formáty sú vnútorne komprimované a majú podobné vlastnosti ako iné komprimované dáta. Častou miskoncepciou žiakov je aj predstava, že komprimáciou už komprimovaných dát dosiahneme podobnú úsporu, ako v prípade nekomprimovaných dát. Žiaci majú k dispozícii vopred pripravené obrázkové súbory a predikčnú kartu (použijeme nástroj pre formatívne hodnotenie **Predikčná karta**, Tabuľka 5), do ktorej zaznamenávajú svoje predpovede a ich zdôvodnenia. Správnosť svojich predpovedí si prakticky overia a zdôvodnia rozdiely medzi ich predpoveďami a skutočnosťou.

Tabuľka 5 Predikčná karta k téme Komprimácia obrázkových súborov

Meno žiaka:		Trieda:		Dátum:	

1. Prezrite si pripravené obrázkové súbory. V tabuľke sú zaznamenané veľkosti príslušných súborov.
2. Uveďte a zdôvodnite svoje predpovede, ako sa zmení veľkosť obrázkových súborov po komprimácii.
3. Skomprimujte obrázkové súbory, zaznamenajte ich veľkosť a vypočítajte úsporu veľkosti.
4. Ak sa líšia vaše predpovede od skutočnosti, zdôvodnite prečo sú vaše predpovede chybné.

SKUTOČNOSŤ stav pred komprimáciou		**PREDPOVEDE očakávané výsledky po komprimácii**		**SKUTOČNOSŤ výsledky po komprimácii**		
typ obrázka	veľkosť (kB)	veľkosť (kB)	**zdôvodnenie**	veľkosť (kB)	úspora veľkosti (%)	**ZDÔVODNENIE ak sa líšia vaše PREDPOVEDE od SKUTOČNOSTI, zdôvodnite prečo**
jednofarebný bmp formát	4955	1500	Na základe skúsenosti s doterajšou kompresiou súborov predpokladám, že sa jednotlivé obrázky skomprimujú na približne tretinu pôvodnej veľkosti	6	99,9	Pre viacfarebné JPEG, PNG, GIF súbory sme dosiahli úsporu blízku 0. V obrázkoch sa opakuje málo častí, a obrázku sú už komprimované. Pre jednofarebné BMP, JPEG, PNG súbory sme dosiahli úsporu cca 90 a viac percent. V obrázku je veľa opakujúcich sa častí. Úspora skoro 100 % je pri jednofarebnom BMP súbore, lebo súbor nie je komprimovaný.
viacfarebný bmp formát	4955	1500		3600	27,3	
jednofarebný jpeg formát	11	4		1	90,9	
viacfarebný jpeg formát	226	70		225	0,4	
jednofarebný png formát	8	3		1	87,5	
viacfarebný png formát	4029	1200		4008	0,5	
jednofarebný gif formát	4	2		3	25	
viacfarebný gif formát	819	250		815	0,5	

Zistené veľkosti súborov sa môžu líšiť v závislosti od obsahu obrázkov a použitého komprimačného programu. Celkové vyhodnotenie to však ovplyvní len minimálne.

Špecifiká aritmetiky počítača

Na tejto hodine sa zaoberáme problematikou binárnej reprezentácie podmnožiny racionálnych (desatinných) čísel v dátovom type float. Žiaci majú predstavu, že každé reálne číslo je možné reprezentovať v počítači. Niektorí žiaci majú skúsenosť, že príliš veľké čísla už nie je možné v počítači reprezentovať. Žiakom chceme ukázať, že pri programovaní výpočtov využívajúcich premenné typu float musíme brať do úvahy špecifiká typu float, a to hlavne obmedzenie rozsahu desatinných čísel (s aspektmi pretečenia a podtečenia) a obmedzenú presnosť výpočtu len na určitý počet platných číslic. Po absolvovaní tejto hodiny by žiaci mali dospieť k nasledovným poznatkom:

- hodnoty typu float sú podmnožinou reálnych čísel,
- hodnoty typu float majú obmedzený počet platných číslic (určených prípustnou veľkosťou mantisy),
- hodnoty typu float sú obmedzené zhora aj zdola prípustnou veľkosťou exponentu,
- čísla, ktoré majú v desiatkovej sústave konečný zápis, môžu mať v dvojkovej sústave nekonečný periodický zápis (a tiež to platí naopak),
- každá vstupná hodnota typu float sa nahradí najbližším dvojkovým číslom s prípustnou veľkosťou mantisy a exponentu.
- pri výpočtoch s desatinnými číslami musíme rátať s obmedzeným počtom platných číslic a obmedzeným číselným rozsahom a tým aj s nepresnými a nesprávnymi výsledkami vzhľadom k matematickej aritmetike (napr. neplatnosť asociatívneho zákona, súčet dvoch kladných čísel je menší ako jeden z jeho sčítancov, podielom dvoch kladných čísel je 0),
- pri programovaní výpočtových problémov s desatinnými číslami uloženými v type `float` treba byť obozretní a zvažovať, či nám bude postačovať rozsah čísel a počet platných číslic,
- pri testoch hodnoty netestujeme zhodu s konkrétnou hodnotou, ale testujeme príslušnosť k intervalu,
- niektoré problémy pri práci s typom float môžeme eliminovať použitím iných dátových typov (napr. `int`), použitím modulov (napr. decimal, fractions), prípadne použitím symbolickej manipulácie.

Ďalej uvádzame len výťah z bádateľsky orientovanej hodiny využívajúcej učebný cyklus 5E.

V časti Zapojenie žiaci vyplnia prvú časť tabuľky (použijeme nástroj pre formatívne hodnotenie **Predtým a potom**, Tabuľka 6). Druhú časť tabuľky žiaci vyplnia na konci vyučovania v časti Vyhodnotenie.

Tabuľka 6 Karta Predtým a potom k téme Špecifiká aritmetiky počítača

Na začiatku hodiny doplňte v stĺpci Predtým k jednotlivým tvrdeniam či platia alebo nie. Na konci hodiny doplňte v stĺpci Potom k jednotlivým tvrdeniam či platia alebo nie a tiež váš komentár ak nie sú rovnaké vaše odpovede v stĺpcoch Predtým a Potom.			
Predtým	Tvrdenie	Potom	Komentár
áno – nie	Uvedený program **sa zacyklí** (neskončí).	áno – nie	

	`n = 1`
	`while n > 0:`
	`    n = n / 2`		

áno – nie	Každé **reálne číslo** sa dá **presne uložiť** do pamäti pomocou niektorého neceločíselného dátového typu (napr. `float`).	áno – nie	
áno – nie	Každé **desatinné číslo** s konečným desatinným rozvojom sa dá **presne uložiť** v pamäti pomocou niektorého neceločíselného dátového typu (napr. `float`).	áno – nie	
áno – nie	**Desatinné čísla** uložené v dátovom type `float` **sú obmedzené** len na určitý **interval** hodnôt.	áno – nie	
áno – nie	Niektoré **nenulové desatinné čísla** (`float`) sa uložia v pamäti ako **nula.**	áno – nie	
áno – nie	**Rozdiely hodnôt** ľubovoľných dvojíc susedných čísel uložených v type `float` **sú rovnaké.**	áno – nie	
áno – nie	Môže nastať situácia, že pre nejaké hodnoty desatinných čísel `a`, `b`≠`0` vráti príkaz `print(a + b == a)` hodnotu **`True`**?	áno – nie	
áno – nie	Môže nastať situácia, že pre nejaké hodnoty desatinných čísel `a`, `b`, `c` vráti príkaz `print((a + b) + c == a + (b + c))` hodnotu **`False`**?	áno – nie	

V časti Skúmanie žiaci riešia niekoľko úloh, z ktorých uvádzame nasledovné dve ukážky. V prvej úlohe žiaci vyplnia tabuľku (použijeme nástroj pre formatívne hodnotenie **Predikčná karta**, Tabuľka 7). Svoje predpovede si žiaci overia praktickou činnosťou v konzole interpretera Pythonu a stručne zhodnotia rozdiely medzi predpoveďami a skutočnosťou.

Tabuľka 7 Predikčná karta k téme Špecifiká artimetiky počítača

Uveďte a zdôvodnite svoje predpovede výsledkov týchto výrazov. Ak ste to urobili, overte si svoje predpovede pomocou konzoly nainštalovaného interpretera Pythonu, resp. online konzoly (https://www.python.org/shell/) a zapíšte a zhodnoťte ich do posledných dvoch stĺpcov tabuľky.

Výraz	Predpoveď výsledku	Zdôvodnenie vašej predpovede	Skutočný výsledok	Bola správna vaša predpoveď?
0.1 + 0.2 - 0.3				áno / nie
(0.1 + 0.2) + 0.3				áno / nie
0.1 + (0.2 + 0.3)				áno / nie
1 + 1e16 > 1e16				áno / nie

Očakávali ste takéto výsledky? Stručne zdôvodnite rozdiely medzi vašimi očakávaniami a skutočnosťou.

Skutočnými výsledkami uvedených výrazov sú:

- 5.551115123125783e-17

- 0.6000000000000001
- 0.6
- False

Až na tretí výsledok, ostatné výsledky sú pre žiakov prekvapivé. V prvom riadku žiaci očakávajú výsledok 0. Rôzne výsledky v druhom a treťom riadku ukazujú, že pri týchto výpočtoch neplatí asociatívny zákon. V poslednom výraze to vyzerá, že sa číslo 1 vôbec nepričítalo k číslu 1e16. Tieto pre žiakov prekvapivé výsledky môžu byť silnou motiváciou pre ďalšie študovanie špecifík aritmetiky počítača.

V druhej úlohe žiaci vyplnia tabuľku (použijeme nástroj pre formatívne hodnotenie **Karta k tvorbe záverov**, Tabuľka 8). Žiaci na základe čiastkových súdov uvedených v stĺpci Z toho usudzujem, že logicky vyvodia záver k rozsahu desatinných čísel.

Tabuľka 8 Karta k tvorbe záverov k téme Špecifiká artimetiky počítača

Zadajte v konzole nasledovné výrazy. Výsledky výrazov spolu so zdôvodnením zapíšte do tabuľky.		
Výraz	Výsledok	Z toho usudzujem, že
1.7e308		
1.8e308		
2*1.7e308		
2**1024		
2.0**1024		
Na základe uvedených súdov logicky vyvodzujem nasledovný záver k **rozsahu desatinných čísel**:		

Výsledkami uvedených výrazov sú:

- 1.7e308
- inf
- inf
- 179769313486231590772930519078902473361797697894230657273430081157732675805500963132708477322407536021120113879871393357658789768814416622492847430639474124377767893424865485276302219601246094119453082952085005768838150682342462881473913110540827237163350510684586298239947245938479716304835356329624224137216
- Traceback (most recent call last): File "<input>", line 1, in <module> OverflowError: (34, 'Result too large')

Výsledkom prvého výrazu je očakávaná hodnota $1.7x10^{308}$, čo je stále v prípustnom rozsahu pre typ float. Výsledkami druhého a tretieho výrazu je hodnota označená ako inf (nekonečno), v tomto prípade bol prekročený prípustný rozsah pre typ float. Výsledok štvrtého výrazu je veľké prirodzené číslo typu int. Posledný výraz sa vyhodnotí ako chyba pretečenia, čo je spôsobený tým, že základ mocniny je typu float, a teda aj výsledok bude typu float, ktorý je už obmedzený rozsahom.

Komentár k formulácii záveru:

- Hodnoty typu float majú obmedzený rozsah, číslo väčšie ako cca 1.7e308 sa označuje ako inf (angl. infinity) alebo výsledok výpočtu nad rozsah chybovou hláškou OverflowError: (34, 'Result too large').
- Pre hodnoty typu int platia iné obmedzenia ako pre typ float. Pre zistenie konkrétnych obmedzení typu int je potrebné realizovať ďalšie skúmanie.

Na konci vyučovania v časti Vyhodnotenie sa žiaci vrátia k tabuľke zo začiatku vyučovania (Tabuľka 6) a vyplnia jej druhú časť. Mali by dospieť k nasledovným tvrdeniam uvedeným v tabuľke (Tabuľka 9). V prípade nesprávnych rozhodnutí by mali vysvetliť prečo ich rozhodnutia o pravdivosti výrokov neboli správne.

Tabuľka 9 Časť karty Predtým a potom k téme Špecifiká aritmetiky počítača so správnym označením pravdivostnej hodnoty výrokov

Tvrdenie	Platí

Uvedený program **sa zacyklí** (neskončí).
`n = 1`
`while n > 0:`
`    n = n / 2`
	nie

Každé **reálne číslo** sa dá **presne uložiť** do pamäti pomocou niektorého neceločíselného dátového typu (napr. `float`).	nie
Každé **desatinné číslo** s konečným desatinným rozvojom sa dá **presne uložiť** v pamäti pomocou niektorého neceločíselného dátového typu (napr. `float`).	nie
Desatinné čísla uložené v dátovom type `float` **sú obmedzené** len na určitý **interval** hodnôt.	áno
Niektoré **nenulové desatinné čísla** (`float`) sa uložia v pamäti ako **nula.**	áno
Rozdiely hodnôt ľubovoľných dvojíc susedných čísel uložených v type `float` **sú rovnaké.**	nie
Môže nastať situácia, že pre nejaké hodnoty desatinných čísel `a`, `b`≠`0` vráti príkaz `print(a + b == a)` hodnotu **`True`**?	áno
Môže nastať situácia, že pre nejaké hodnoty desatinných čísel `a`, `b`, `c` vráti príkaz `print((a + b) + c == a + (b + c))` hodnotu **`False`**?	áno