



UNIVERZITA KONŠTANTÍNA FILOZOFA V NITRE
FAKULTA PRÍRODNÝCH VIED A INFORMATIKY

ZÁKLADY ALGORITMIZÁCIE A PROGRAMOVANIA

Programovanie v prostredí Scratch

Gabriela Lovászová

NITRA 2023

ZÁKLADY ALGORITMIZÁCIE
A PROGRAMOVANIA: PROGRAMOVANIE
V PROSTREDÍ SCRATCH

GABRIELA LOVÁSZOVÁ

2023

Základy algoritmizácie a programovania: Programovanie v prostredí Scratch

Edícia Prírodovedec č. 821

Autor:

doc. RNDr. Gabriela Lovászová, PhD.

Recenzent:

PaedDr. Viera Michaličková, PhD.

(c) 2023 Univerzita Konštantína Filozofa v Nitre

Publikácia bola vytvorená v rámci projektu *001UKF-2-1/2022 Zvyšovanie kvality prípravy budúcich učiteľov matematiky, fyziky, chémie, informatiky, anglického jazyka, slovenského jazyka a techniky formou doplňujúceho pedagogického štúdia a rozširujúceho štúdia na UKF v Nitre.*

ISBN 978-80-558-2055-2

OBSAH

Úvod	6
Lekcia 1	7
1.1 Programovacie prostredie Scratch	7
1.2 Príkazy a scenáre	10
1.3 Cyklus – opakovanie bloku príkazov	11
1.4 Vlastné príkazy	12
1.5 Udalosti	14
1.6 Uloženie a zdieľanie projektu	15
Lekcia 2	16
2.1 Údaje ako parametre príkazov a vlastnosti objektov	16
2.2 Operácie s údajmi	19
2.3 Premenné	21
2.4 Logické výrazy a podmienené príkazy	23
2.5 Vstupy a výstupy	26
Lekcia 3	29
3.1 Rozšírenie Pero	29
3.2 Kreslenie, farba a hrúbka pera	30
3.3 Príkazy s parametrami	34
3.4 Pečiatkovanie	39
Lekcia 4	41
4.1 Viac postáv v projekte	41
4.2 Animácia a zvuky	42
4.3 Komunikácia objektov v projekte	44
Lekcia 5	49
5.1 Kategórie príkazov	49
5.2 Riadenie výpočtu	50
5.3 Podprogramy	53
5.4 Práca s údajmi	54
5.5 Práca s objektmi	60
Záver	62

Literatúra.....	63
Prílohy.....	64

ÚVOD

Skriptum je určené pre študentov učiteľstva informatiky, rozširujúceho štúdia a doplnkového pedagogického štúdia. Jeho cieľom je prezentovať základné koncepty z programovania prostredníctvom programovacieho prostredia Scratch určeného pre detského používateľa.

Študenti učiteľstva informatiky rozumejú, že informatika je vedecká a odborná disciplína o automatizovanom spracovaní informácií uchovávaných vo forme digitálnych dát, ktoré sa deje podľa algoritmov v programoch. Majú znalosti o tvorbe algoritmov a vedia riešiť problémy algoritmicky. Ovládajú aspoň jeden profesionálny programovací jazyk, napríklad Python, a vedia pomocou neho zapisovať algoritmické riešenia problémov. Pomocou vlastných programov vedia získať vlastné výsledky riešenia problémov, a tiež tvoriť aplikácie s príjemným používateľským rozhraním. Rozumejú, ako prebieha vývoj aplikácie, a vedia pri tvorbe aplikácie používať vývojové prostredie.

Okrem týchto odborných vedomostí a zručností má mať budúci učiteľ informatiky dôležitú schopnosť priblížiť podstatu algoritmizácie, programovania a tvorby aplikácií svojim žiakom na úrovni primeranej ich veku a schopnostiam, a vzbudiť záujem žiakov o programovanie.

V študijnom texte predstavujeme vývojové prostredie Scratch, ktoré je vhodné na programovanie pre deti vo veku žiakov základnej školy. Text aj úlohy sú určené pre budúceho učiteľa informatiky tak, aby

- spoznal možnosti programovacieho prostredia Scratch,
- získal zručnosť v programovaní v prostredí Scratch,
- vedel identifikovať všeobecné koncepty programovania v prostredí Scratch a zosúladiť ich so svojimi vedomosťami z programovania v iných programovacích jazykoch a vývojových prostrediach,
- vedel posúdiť výhody programovania v detskom programovacom prostredí pre vyučovacie ciele,
- bol pripravený navrhovať didaktické transformácie učiva a metodické postupy pre konkrétne cieľové skupiny žiakov,
- chápal programovanie ako príležitosť na rozvoj tvorivosti.

S teoretickými konceptmi vyučovania programovania a ich praktickými aplikáciami pri navrhovaní metodických vyučovacích postupov sa študenti učiteľstva informatiky stretnú v rámci svojho štúdia v predmete Didaktika informatiky a na pedagogických praxiach. Metodické postupy, ako vyučovať žiakov programovanie v prostredí Scratch, učebnice, pracovné listy a iné materiály určené pre deti nájde čitateľ v odkazoch na literatúru.

LEKCIA 1

Cieľom tejto lekcie je:

- zoznámiť sa s vývojovým prostredím Scratch,
- pochopiť objektovú štruktúru projektu vytvoreného v prostredí Scratch osvojiť si terminológiu – objekty typu scéna a postava, vlastnosti objektov,
- poznať príkazy na ovládanie pohybu postavy,
- vedieť vykonať príkaz s parametrom interaktívne vo vývojovom prostredí,
- zostavovať scenáre ako postupnosti príkazov,
- poznať a vedieť používať príkaz cyklu s daným počtom opakovaní,
- riešiť jednoduché algoritmické úlohy s využitím postupnosti príkazov a cyklu s daným počtom opakovaní,
- vytvoriť nový blok príkazov bez parametra a vedieť ho použiť pri riešení úloh,
- spúšťať scenáre interaktívne vo vývojovom prostredí alebo ako reakcie na udalosti pri kliknutí na postavu, pri stlačení klávesu, pri kliknutí na zástavku,
- vedieť uložiť projekt do súboru a publikovať projekt v cloude prostredia Scratch.

Motivačnou úlohou v tejto lekcií je programovanie pohybu autíčka v meste. Auto predstavuje objekt, ktorý vieme ovládať príkazmi. Mapa mesta obsahuje cesty a rôzne objekty a tvorí prostredie pre formulovanie rôznych úloh pre pohyb auta.

1.1 PROGRAMOVACIE PROSTREDIE SCRATCH

Scratch je on-line programovacie prostredie dostupné na webovej adrese <https://scratch.mit.edu/>. Na jeho používanie je potrebný webový prehliadač a prístup na internet. Programovanie v prostredí Scratch je možné bez alebo s vytvorením používateľského účtu. Vytvorenie účtu a neanonymné používanie prostredia Scratch prináša výhody v podobe automatického ukladania projektov a vytvárania používateľského portfólia v cloude, možnosti publikovania projektov a komunikácie v komunite používateľov Scratchu (Scratcherov).

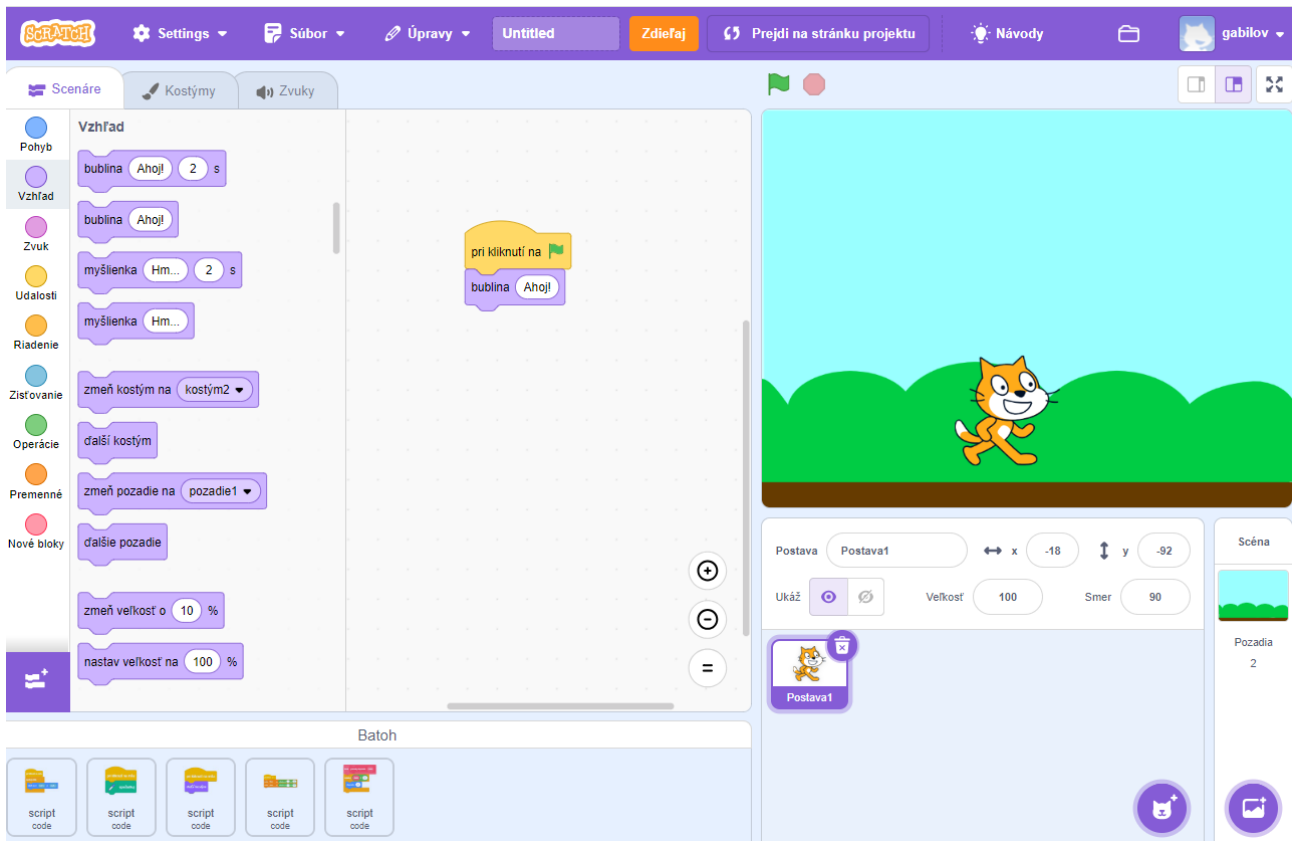
Scratch je možné používať aj ako desktopovú aplikáciu bez on-line pripojenia, vtedy je potrebná inštalácia v počítači. Inštalčné programy vývojového prostredia Scratch sú dostupné zadarmo pre rôzne softvérové platformy na stránkach Scratch.

On-line vývojové prostredie Scratch otvoríme voľbou **Vytvor** na domovskej stránke prostredia Scratch.



Okno vývojového prostredia sa skladá z troch hlavných častí:

- editor,
- prehliadač objektov,
- okno s výstupom.



Editor v ľavej polovici pracovnej plochy má tri karty:

- **Scenáre** – blokový editor – tu sa editujú programy, voláme ich aj scenáre alebo skripty.
- **Kostýmy/Pozadia** – grafický editor – tu sa edituje grafický vzhľad objektov: postáv (vzhľad postavy voláme kostým) a scény (vzhľad scény voláme pozadie).
- **Zvuky** – zvukový editor – tu sa editujú zvuky, ktoré môžu vydávať objekty (postavy alebo scéna).

V spodnej časti editora je **Batoh**, ktorý slúži ako schránka na ukladanie a prenášanie častí programov medzi scenármi pre rôzne objekty, aj medzi rôznymi projektmi.

Prehliadač objektov sa nachádza v spodnej časti pravej polovice pracovnej plochy. Tu vidíme štruktúru projektu, z akých objektov sa skladá. Objekty sú dvoch typov:

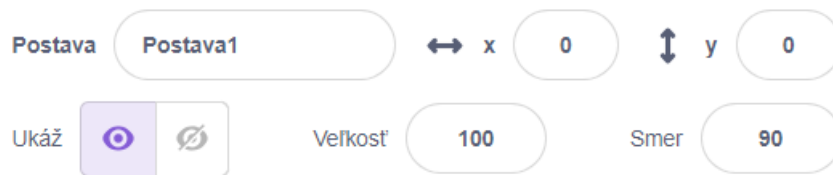
- **Postavy** – tu vidíme prehľad všetkých postáv a vlastnosti práve vybranej postavy.
- **Scéna** – scéna je v projekte vždy len jedna.

Výstup projektu sa nachádza v hornej časti pravej polovice pracovnej plochy. Tu môžeme sledovať, ako funguje náš projekt. Tlačidlo so zelenou zástavkou slúži na spustenie, červený kruh na zastavenie behu programu. Veľkosť časti okna s výstupom sa dá meniť:

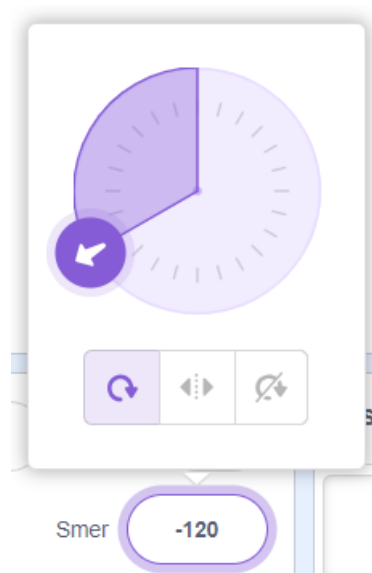
- malá veľkosť,
- stredná veľkosť,
- celé okno.

ÚLOHY

1. V novovytvorenom projekte je jedna postava kocúra na scéne s prázdnyim pozadím. Preskúmajte projekt vo vývojovom prostredí.
2. Zmeňte:



- meno postavy kocúra,
- súradnice kocúra ťahaním myšou vo výstupnom okne alebo prepísaním súradníc v Prehliadači objektov,
- veľkosť kocúra prepísaním čísla v Prehliadači objektov,
- skrytie/ukážte kocúra pomocou ikony oka,
- smer kocúra v Prehliadači objektov prepísaním čísla alebo posúvaním šípky na terčíku myšou,



3. Zistite na karte Kostýmy v časti editora, koľko kostýmov má postava kocúra.
4. Zmeňte kostým kocúra, pridajte nový kostým z ponuky kostýmov v prostredí Scratch.



5. Má postava kocúra nejaké zvuky? Zistite na karte Zvuky.
6. Kliknite na objekt Scéna v Prehliadači objektov. Preskúmajte jej pozadia a zvuky.
7. Pridajte nové pozadie scény z ponuky pozadí prostredia Scratch.

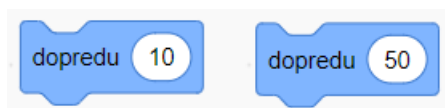


1.2 PRÍKAZY A SCENÁRE

V Prehliadači objektov si zvolíme objekt (postavu alebo scénu), ktorému chceme zadávať príkaz. V Editore príkazov sa objaví **ponuka príkazov** pre tento objekt. Príkazy majú tvar blokov skladačky puzzle. Ponuka príkazov je štruktúrovaná do kategórií, ktoré sa odlišujú farbou bloku. Po kliknutí myšou na blok sa príkaz vykoná. Výsledok môžeme sledovať vo výstupnom okne.

Niektoré príkazy majú meniteľné **parametre**, ktoré sa dajú pred vykonaním príkazu v bloku editovať.

Napríklad pre postavy máme v ponuke príkazov v kategórii Pohyb príkaz *dopredu*, ktorý môžeme vykonať s rôznymi hodnotami parametra:



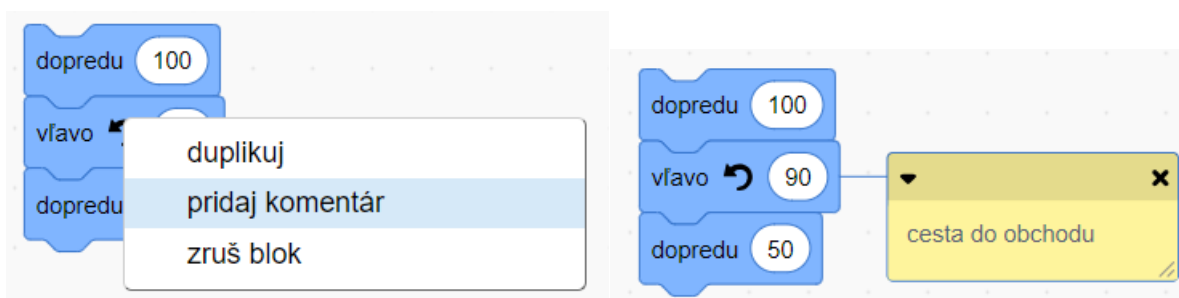
Vyskúšajme niektoré príkazy pre postavu kocúra – po kliknutí myšou na blok príkazu sledujme jeho efekt.

Scenár zostavujeme v okne Editor z **príkazov**. Príkaz z ponuky presunieme do editora ťahaním myšou. V editore môžeme pospájať viac príkazov do **postupnosti** a vykonať ju naraz kliknutím myši na celý blok.

Scenár z troch príkazov spojených do postupnosti:



Na prehľadnenie scenára môžeme použiť **komentáre**. V komentári si poznačíme, čo daný scenár alebo časť scenára robí. Kontextovú ponuku na pridanie komentára vyvoláme stlačením pravého tlačidla myši.



ÚLOHY

Otvorte projekt Mesto (<https://scratch.mit.edu/projects/867103226/>). V projekte je scéna s pozadím mesta, ktoré bolo vytvorené pomocou iného zverejneného projektu City Builder 4.0 Sandbox (<https://scratch.mit.edu/projects/327571645>) a postava s kostýmom auta. Vytvorte scenáre pre postavu auto:

1. Auto prejde z domovskej pozície pri najvyššom sivom paneláku (príkaz *domov* v skupine Nové bloky) do banky. Použite príkaz *dopredu* zo skupiny Pohyb alebo *pomaly dopredu* zo skupiny Nové bloky. Aký je rozdiel?
2. Auto prejde z domovskej pozície na úrad.
3. Auto prejde z domovskej pozície do obchodu.
4. Pridajte komentáre k scenárom, ktoré opisujú, čo scenár robí.

1.3 CYKLUS – OPAKOVANIE BLOKU PRÍKAZOV

Na opakovanie bloku príkazov daný počet krát slúži príkaz *opakuj* z ponuky príkazov v skupine Riadenie (oranžové). Takéto opakovanie časti programu voláme **cyklus**.



Blok cyklu má tvar písmena C. Dvnútra bloku sa môžu vkladať príkazy, ktoré sa majú opakovať. Tvoria **telo cyklu**. Šírka oranžového bloku cyklu sa bude prispôsobovať veľkosti tela. Počet opakovaní sa nastavuje parametrom.

V tomto cykle sa 24-krát opakuje dvojica príkazov *dopredu 10* a *vpravo 15*.



Pracujeme s projektom Mesto. Zostavme pre postavu auto takýto scenár: Auto presuňme z domovskej pozície na križovatku na rohu námestia. Odtiaľ obíde štvorcové námestie a skončí na pozícii ako je vyznačené na obrázku.



V scenári vieme identifikovať 4-krát sa opakujúcu dvojicu príkazov *pomaly dopredu 160* a *vpravo 90*. Zápis môžeme skrátiť s využitím cyklu *opakuj*:



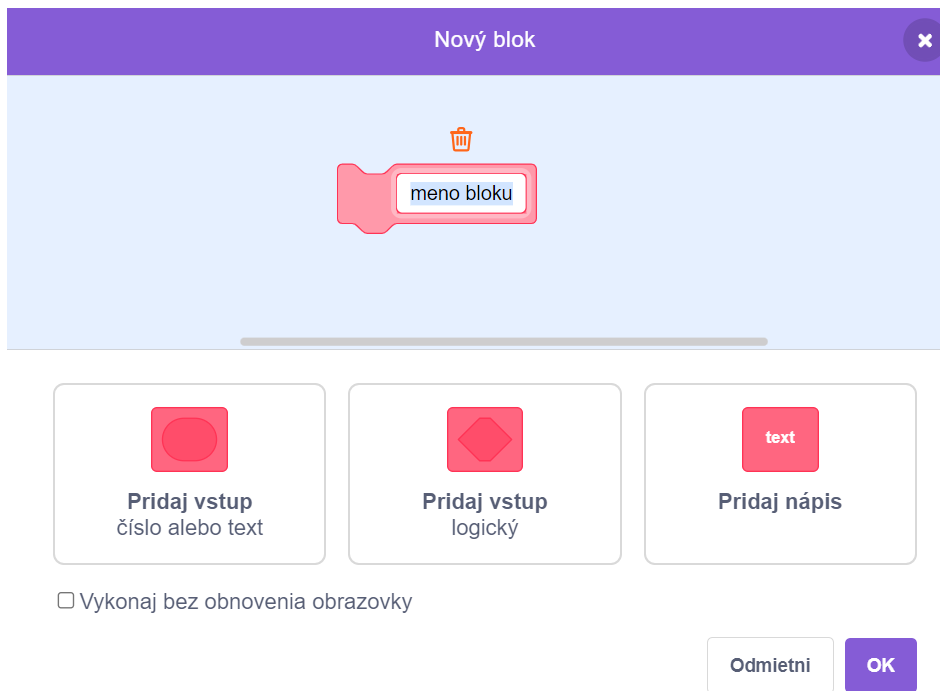
ÚLOHY

1. Upravte scenár tak, aby auto obišlo námestie dvakrát, trikrát.
2. Upravte scenár tak, aby auto jedenkrát obišlo celé námestie a skončilo pri banke.
3. Upravte scenár tak, aby auto dvakrát obišlo celé námestie a vrátilo sa domov.
4. Upravte scenár tak, aby auto trikrát obišlo celé námestie a potom skončilo pri obchode.

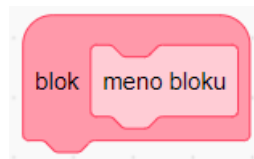
1.4 VLASTNÉ PRÍKAZY

Príkazy v ponuke príkazov môžeme rozšíriť o **vlastné príkazy**. Ľubovoľný scenár môžeme nazvať menom a vytvoriť tak nový príkaz, ktorý bude vykonávať daný scenár.

V skupine Nové bloky stlačíme tlačidlo „Nový blok“. V dialógovom okne do pripraveného červeného bloku napíšeme meno nového príkazu.



Po potvrdení (OK) sa nový blok zobrazí v ponuke v skupine *Nové bloky* a v editore scenárov sa objaví **hlavička** nového príkazu s kľúčovým slovom *blok* a menom nového príkazu. Hlavička predstavuje začiatok nového bloku príkazov a má tvar, ku ktorému sa dajú pripájať bloky len na koniec.



Vytvoríme nový príkaz *okolo doprava*, ktorý z ľubovoľného rohu obide námestie v pravotočivom smere:



V ponuke príkazov v kategórii *Nové bloky* pribudol nový príkazový blok *okolo doprava*, ktorý môžeme používať v našich programoch rovnako ako iné bloky z ponuky.

Vytvoríme scenár s využitím nového bloku *okolo doprava*, v ktorom auto vyjde z domovskej pozície, dvakrát zakrúži okolo námestia a presunie sa k banke.



Do kategórie Nové bloky patria aj príkazy *pomaly dopredu* a *domov*.

ÚLOHY:

Pokračujeme s projektom Mesto.

1. Použite príkaz *okolo doprava* v riešení úloh z predchádzajúcej kapitoly.
2. Vytvorte nový príkaz *okolo doľava*, ktorý z ľubovoľného rohu obíde námestie v ľavotočivom smere.
3. Vytvorte vlastný príkaz *do obchodu*, ktorým auto prejde z domovskej pozície k obchodu na druhej strane rieky.
4. Vytvorte vlastný príkaz *k lesoparku*, ktorým auto prejde z domovskej pozície k lávke cez rieku do lesoparku.
5. Vytvorte vlastný príkaz *city tour*, ktorým auto prejde z domovskej pozície k lesoparku, tam sa zastaví na 3 sekundy, potom sa vráti do mesta, raz obíde námestie a skončí pri bufete pri jazere. Použite vlastné príkazy, kde sa dá. Na čakanie použite príkaz *čakaj* z kategórie Riadenie.

1.5 UDALOSTI

Interaktívny projekt interaguje počas vykonávania programu s používateľom, napríklad reaguje na vstupy z klávesnice alebo myši. V moderných programovacích jazykoch sa takáto interaktivita zabezpečuje prostredníctvom reakcií na **udalosti**.

V kategórii príkazov Udalosti sa nachádzajú žlté bloky, ktoré podobne ako hlavička nového bloku príkazov majú tvar, ku ktorému sa dajú pridávať príkazy len zdola. Scenáre pripojené k udalostnému bloku sa vykonajú, keď počas vykonávania programu nastane príslušná udalosť.

Príklady udalostných blokov: Bloky uvádzajú scenáre, ktoré sa majú vykonať *pri kliknutí* myšou na zelenú zástavku, *pri stlačení* medzerníka na klávesnici (medzerník je parameter, dá sa nastaviť aj iný kláves z klávesnice), *pri kliknutí* myšou na postavu, ktorej je scenár určený.



Naprogramujeme reakciu na udalosť auta *pri kliknutí na mňa* tak, aby auto zatrúbilo. Použijeme blok príkazu *zahraj zvuk* z kategórie príkazov Zvuky. Príkaz má jeden parameter, ktorým sa udáva meno zvuku, ktorý chceme prehrať. Zvuk sa vyberá z ponuky zvukov, ktoré má postava k dispozícii. Môžeme si ich pozrieť, editovať, pridávať, mazať na karte Zvuky.

Trúbiť autom môžeme aj pomocou klávesnice tak, že zahrание zvuku klaksóna vykonáme pri stlačení klávesu. Udalostný blok pri stlačení má ako parameter kláves, na stlačenie ktorého má auto zareagovať.



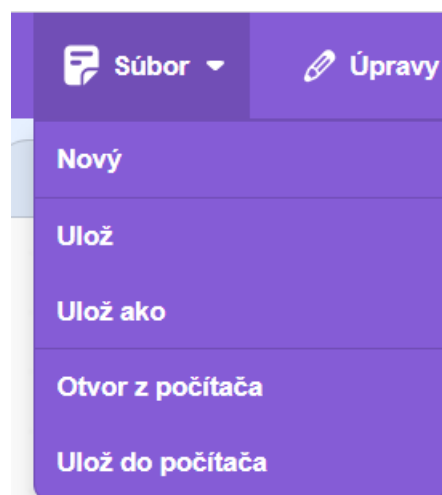
ÚLOHY:

Pracujeme s projektom Mesto.

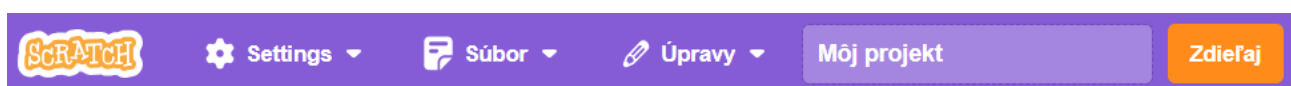
1. Pri kliknutí na zelenú zástavku nastavte auto do domovskej pozície. Vyskúšajte reakciu kliknutím na zelenú zástavku. Rovnakú akciu urobte aj na stlačenie klávesu „d“.
2. Pri stlačení klávesu „o“ auto prejde z domovskej pozície do obchodu.
3. Pri stlačení klávesu „b“ auto prejde z domovskej pozície do banky.
4. Pri stlačení klávesu „j“ auto prejde z domovskej pozície k jazeru.
5. Pri stlačení klávesu „u“ auto prejde z domovskej pozície k úradu.
6. Pri stlačení klávesu „l“ auto prejde z domovskej pozície k lesoparku.

1.6 ULOŽENIE A ZDIEĽANIE PROJEKTU

Projekt je možné **uložiť do počítača** ako súbor s príponou .sb3 a neskôr znovu otvoriť v prostredí Scratch.



Prihlásení používatelia si môžu svoje projekty ukladať v cloude a zdieľať aj s ostatnými.



LEKCIA 2

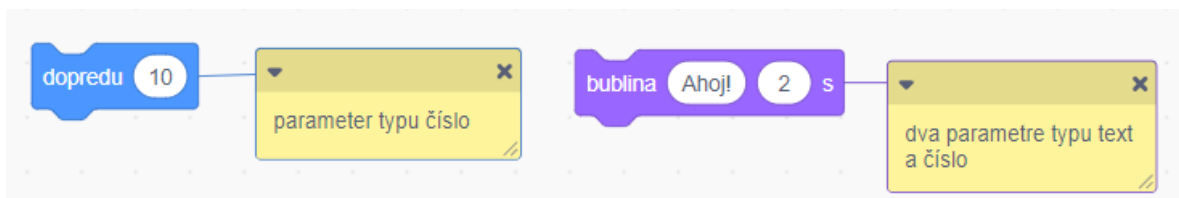
Cieľom tejto lekcie je:

- rozlišovať údaje číselného a textového typu,
- používať číselné a textové údaje ako parametre príkazov,
- nastavovať hodnoty vlastností objektov interaktívne vo vývojovom prostredí a pomocou príkazov,
- zostavovať jednoduché a zložené výrazy s operáciami, číselnými a textovými údajmi,
- používať výrazy ako parametre príkazov a operácií,
- vytvoriť premennú, použiť premennú v príkaze alebo vo výraze, zobrazíť hodnotu premennej vo výstupnom okne,
- zostavovať logické výrazy pomocou relačných a logických operácií,
- poznať podmienené príkazy úplného a neúplného vetvenia, rozumieť ich významu,
- formulovať podmienky pre podmienené príkazy,
- riešiť problémy s využitím premenných, výrazov, podmienených príkazov.

Motivačnou témou tejto lekcie je projekt s postavou počítačky, ktorá rieši počítačské úlohy na základné aritmetické operácie.

2.1 ÚDAJE AKO PARAMETRE PRÍKAZOV A VLASTNOSTI OBJEKTOV

Niektoré príkazy majú **parametre** – vstupné údaje, ktorými programátor špecifikuje význam príkazu. Údaje môžu byť čísla alebo texty. V bloku príkazu má oblasť, v ktorej sa edituje hodnota parametra typu text alebo číslo, oválny tvar. Napríklad:

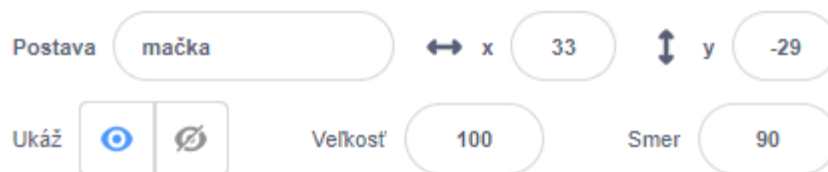


Objekty v projekte (postavy a scéna) majú rôzne **vlastnosti**, ktoré sú uchované vo forme údajov. Napríklad postava má vlastnosti:

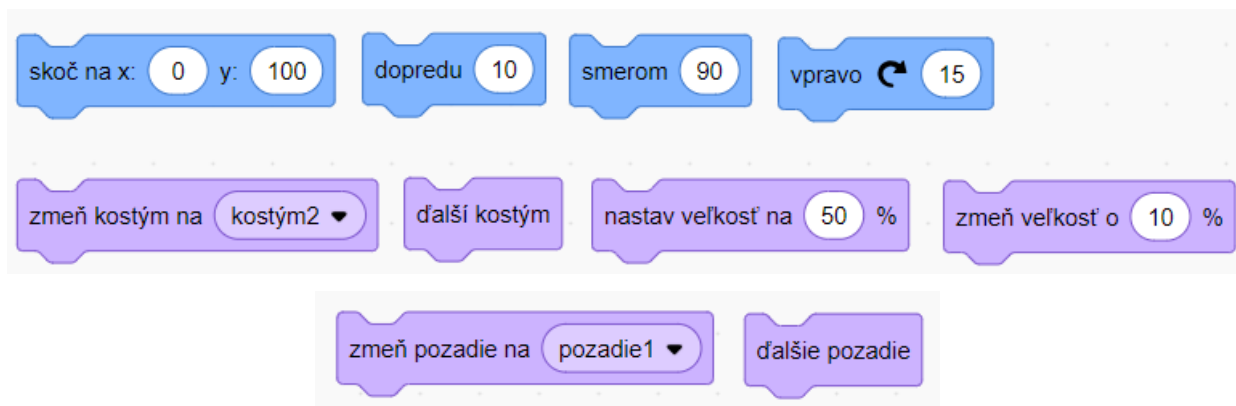
- pozíciu reprezentovanú dvomi číselnými údajmi: súradnicou x, súradnicou y,
- smer reprezentovaný číselnou hodnotou v stupňoch,
- veľkosť reprezentovanú číselným údajom v percentách,
- kostým reprezentovaný buď poradovým číslom v zozname kostýmov, alebo menom kostýmu.

Tieto vlastnosti vieme nastaviť vo vývojovom prostredí, ale aj meniť ich hodnoty pomocou príkazov a zisťovať ich hodnoty pomocou operácií.

Nastavenie hodnôt vlastností postavy vo vývojovom prostredí:



Príklady príkazov, ktoré menia hodnoty vlastností postavy a scény:

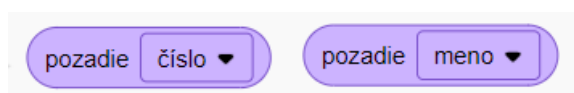


Bloky, ktoré predstavujú hodnoty vlastností objektov, majú oválny tvar, lebo predstavujú číselný alebo textový údaj.

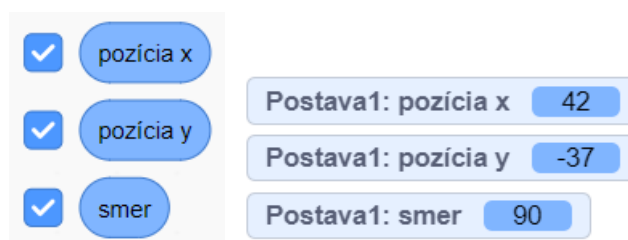
Príklady blokov vlastností postáv:



Príklady blokov vlastnosti scény *pozadie*:



Hodnoty vlastností objektov sa dajú zobrazíť vo výstupnom okne zaškrtnutím zaškrťavacieho políčka pri bloku v ponuke blokov:



Napríklad v projekte Mesto dajme zobrazíť vo výstupnom okne súradnice auta. Premiestnime okienka so zobrazením hodnôt vlastností auta na vhodné miesto na scéne. Sledujme zmenu údajov počas pohybu auta.



ÚLOHY:

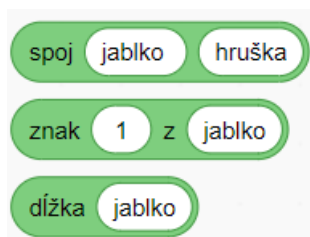
1. Dajte zobrazíť smer auta. Sledujte zmenu údajov počas pohybu auta.
2. Vytvorte nový projekt, v ktorom kocúr Scratch postupne v bublinách oznamuje hodnoty svojich nastavení. Hlásenie odštartujeme stlačením zástavky:
 - Moja x-ová súradnica je...
 - Moja y-ová súradnica je...
 - Pozerám sa smerom...
 - Moja veľkosť je...
 - Mám kostým...
 - s poradovým číslom...

Ukážka časti riešenia:



2.2 OPERÁCIE S ÚDAJMI

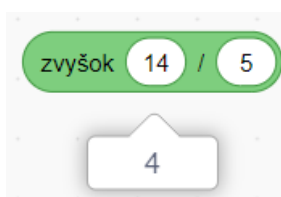
Na prácu s údajmi slúžia bloky zo skupiny **Operácie**. Na prácu s textom sú v prostredí Scratch k dispozícii operácie *spoj* na spojenie dvoch textov do jedného, *znak* na zistenie znaku s daným poradovým číslom v texte, *dĺžka* na zistenie dĺžky textu:



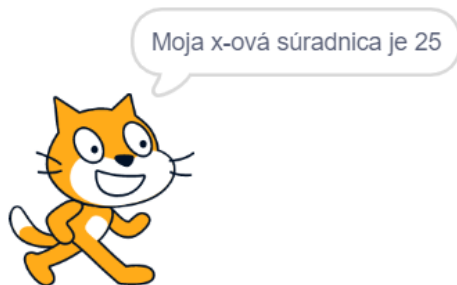
Na prácu s číslami sú v prostredí Scratch k dispozícii aritmetické operácie $+$, $-$, $*$, $/$, operácia *náhodne* na vygenerovanie náhodného čísla z daného intervalu, *zvyšok* na výpočet zvyšku po celočíselnom delení dvoch čísiel, *zaokrúhli* na zaokrúhlenie desatinného čísla na celé a ďalšie matematické operácie (*abs*, *odmocnina*, *sin*, *cos*, atď.):



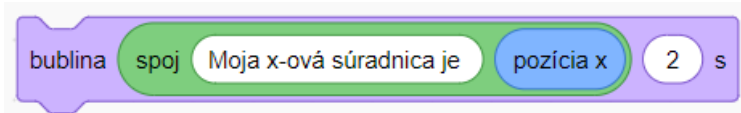
Bloky majú oválny tvar. Ich výsledkom je hodnota typu číslo alebo text a dajú sa vložiť všade tam, kde sa ako údaj očakáva číslo alebo text. Samotné bloky operácií majú vstupné parametre typu číslo alebo text, v ktorých sa dajú editovať konkrétne hodnoty údajov, alebo vkladať ďalšie oválne bloky. Pri kliknutí na oválny blok v editore scenárov sa vyhodnotí operácia a výsledok sa zobrazí v bubline v bubline.



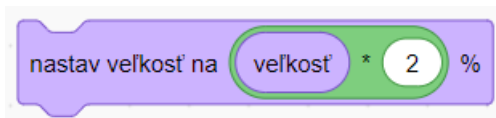
Využime oválne bloky operácií v iných príkazoch. Napríklad upravme hlásenie kocúra o hodnotách svojich vlastností tak, aby hodnota bola v jednej bubline s celou vetou.



Text „Moja x-ová súradnica je“ najprv pomocou operácie *spoj* spojíme s hodnotou vlastnosti *pozícia x* do jedného textu. Tento oválny blok potom vložíme ako parameter príkazu *bublina*.



Zdvojnásobme veľkosť kocúra. Veľkosť kocúra vieme zistiť pomocou oválneho bloku *veľkosť*. Túto hodnotu pomocou operácie *** (násobenie) vynásobíme dvomi. Získanú hodnotu vložíme ako parameter príkazu *nastav veľkosť na*.

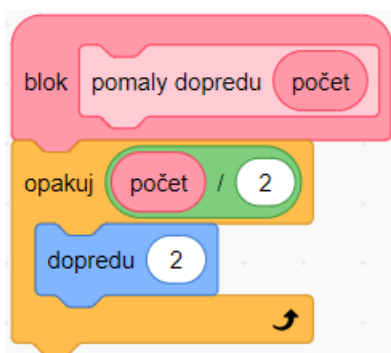


ÚLOHY

1. Otestujte v editore scenárov, čo počítajú bloky z kategórie Operácie pre rôzne hodnoty vstupných parametrov.
2. Upravte aj ostatné hlásenia kocúra o hodnotách svojich vlastností tak, aby hodnota bola v jednej bubline s celou vetou. Pridajte bodku za vetou.
3. Zostavte príkaz, ktorý:
 - zmenší veľkosť kocúra na polovicu,
 - zmení veľkosť kocúra náhodne z intervalu 50 až 200 percent.

Dajte zobraziť hodnotu vlastnosti *veľkosť* a sledujte, ako sa príkazmi mení.

4. Preskúmajte príkaz *pomaly dopredu* v projekte Mesto. Ako funguje animácia pohybu auta? Upravte program tak, aby auto išlo dvakrát rýchlejšie, dvakrát pomalšie. Upravte program tak, aby auto vedelo aj cúvať pri zápornom vstupe.

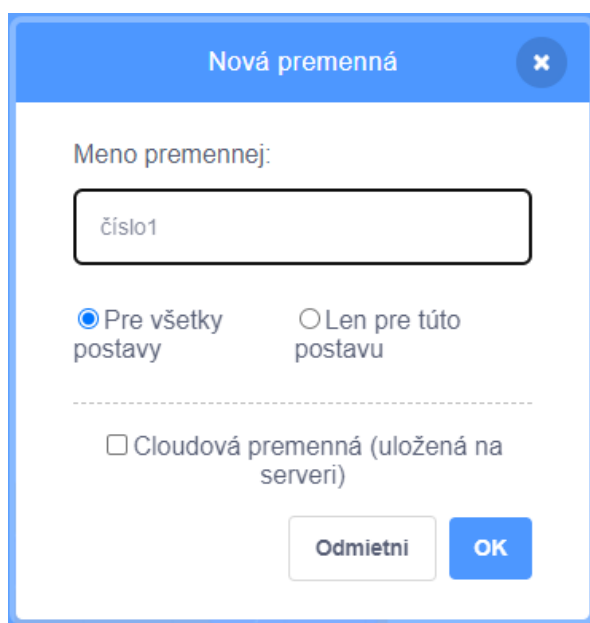


2.3 PREMENNÉ

Premenná je miesto v pamäti na uloženie hodnoty. K premennej prístupujeme prostredníctvom jej **mena** (podobne ako pri vlastnostiach objektov ako *veľkosť*, *kostým*, *smier*, *pozadie* a podobne).

V prostredí Scratch sa premenné vytvárajú v skupine blokov *Premenné* stlačením tlačidla *Nová premenná*. Premenná môže byť **globálna**, teda prístupná všetkým objektom v projekte, alebo **lokálna**, teda prístupná len pre daný objekt (ako je to pri vlastnostiach objektov).

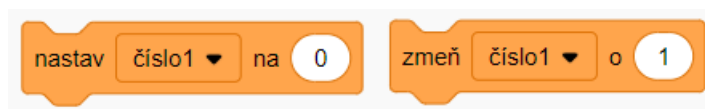
Vytvoríme premennú s menom *číslo1*, ktorá bude prístupná všetkým objektom v projekte:



Zaškrtnutím/odškrtnutím zaškrtačacieho políčka vedľa mena premennej vieme zobrazit/skryť jej **hodnotu** vo výstupnom okne podobne ako pri vlastnostiach objektov (*pozícia x*, *pozícia y*, *smier* a podobne). Ukázať alebo skryť zobrazenie hodnoty premennej vo výstupnom okne vieme tiež pomocou príkazov *ukáž premennú*, *skry premennú*.



Hodnotu premennej vieme meniť príkazmi:



Príkazom *nastav* **priradíme** danej premennej danú hodnotu. Príkazom *zmeň* **meníme** aktuálnu hodnotu premennej pričítaním danej hodnoty.

Hodnota premennej môže byť vo výstupnom okne zobrazená tromi spôsobmi:

- meno premennej a hodnota premennej,



- iba hodnota premennej,

80

- meno premennej, hodnota premennej a posúvač, ktorým sa dá nastavovať číselná hodnota v rozmedzí 0 až 100 interaktívne v bežiacom projekte bez nutnosti použiť príkaz na zmenu hodnoty.

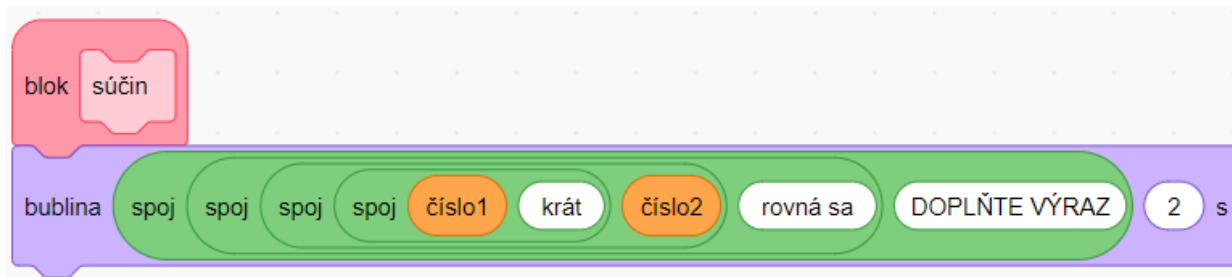


Spôsob zobrazenia premennej sa dá cyklicky meniť dvojklikom. Zobrazenie hodnôt premenných je užitočné na sledovanie výpočtu pri ladení programu (úpravy, opravy, hľadanie chýb), ale môže byť aj súčasťou výsledného výstupu.

Premenné sú v blokovom programovacom jazyku prostredia Scratch reprezentované oválnymi blokmi podobne ako vlastnosti objektov. Tieto oválne bloky môžeme použiť ako parametre operácií a príkazov s číselnými alebo textovými vstupmi.

ÚLOHY

1. Vytvorte projekt Počtárka. Pozadie scény nastavte na čiernu školskú tabuľu (pozadie Chalkboard, ktoré je k dispozícii v ponuke pozadí prostredia Scratch). Pred tabuľou vľavo nech stojí jedna postava s kostýmom dievčaťa Abby (z ponuky kostýmov prostredia Scratch).
2. Vytvorte dve globálne premenné *číslo1* a *číslo2*. Zobrazte posúvač na nastavovanie hodnoty vo výstupnom okne.
3. Vytvorte scenár, v ktorom počtárka celou vetou oznámi súčin hodnôt premenných. Napríklad 12 krát 10 sa rovná 120. Zo scenára vytvorte nový príkazový blok *súčin*. Vyskúšajte počtárku z násobilky. Hodnoty činiteľov zadávajte posúvačom.





4. Upravte vizuál projektu ako na obrázku nižšie. Umiestnite zobrazenie premenných na tabuľu, zvolte zobrazenie hodnoty bez mena premennej. Na tabuľu v pozadí scény dokreslite text „krát“ a zobrazenia hodnôt premenných umiestnite z dvoch strán textu, aby tvorili počtársku úlohu.



5. Pridajte **reakciu scény** na udalosť *pri stlačení* klávesu „n“, v ktorej sa hodnoty premenných nastaví náhodne v rozmedzí -100 až 100.
6. Pridajte **reakciu scény** na udalosť *pri kliknutí na zástavku*, v ktorej sa hodnoty premenných *číslo1* a *číslo2* vynulujú.
7. Pridajte **reakciu počtárky** na udalosť *pri kliknutí na mňa*, v ktorej sa vykoná príkaz *súčin*, teda postava počtárky v bubline celou vetou prezentuje výsledok počtárskej úlohy na tabuli.

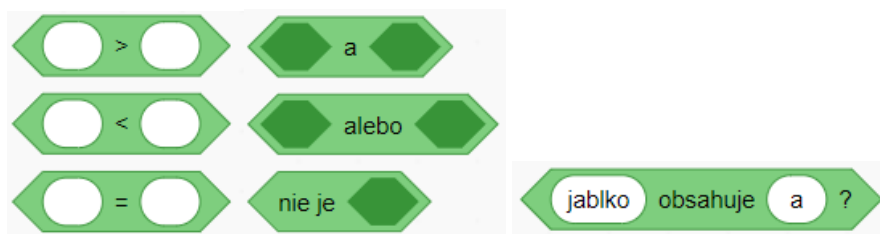
2.4 LOGICKÉ VÝRAZY A PODMIENENÉ PRÍKAZY

Logické výrazy sú výrazy, ktoré majú hodnotu **pravda** (true) alebo **nepravda** (false). Môžu obsahovať:

- relačné operátory $<$, $>$, $=$ na porovnávanie čísiel a reťazcov,

- logické operácie *a* (konjunkcia), *alebo* (disjunkcia), *nie je* (negácia), ktoré operujú s logickými hodnotami,
- operácia *obsahuje*, ktorá zistí, či zadaný prvý text obsahuje druhý text.

Bloky takýchto výrazov majú šesťuholníkový tvar:



Keď v editore scenárov kliknutím vykonáme výpočet šesťuholníkového bloku, zobrazí sa v bubline výsledná hodnota *true* alebo *false* (v angličtine).



Logické výrazy sa používajú v **podmienených príkazoch**, ktoré vykonávajú príkazy, ak je splnená nejaká **podmienka**. Podmienené príkazy nájdeme v skupine príkazov *Riadenie*.

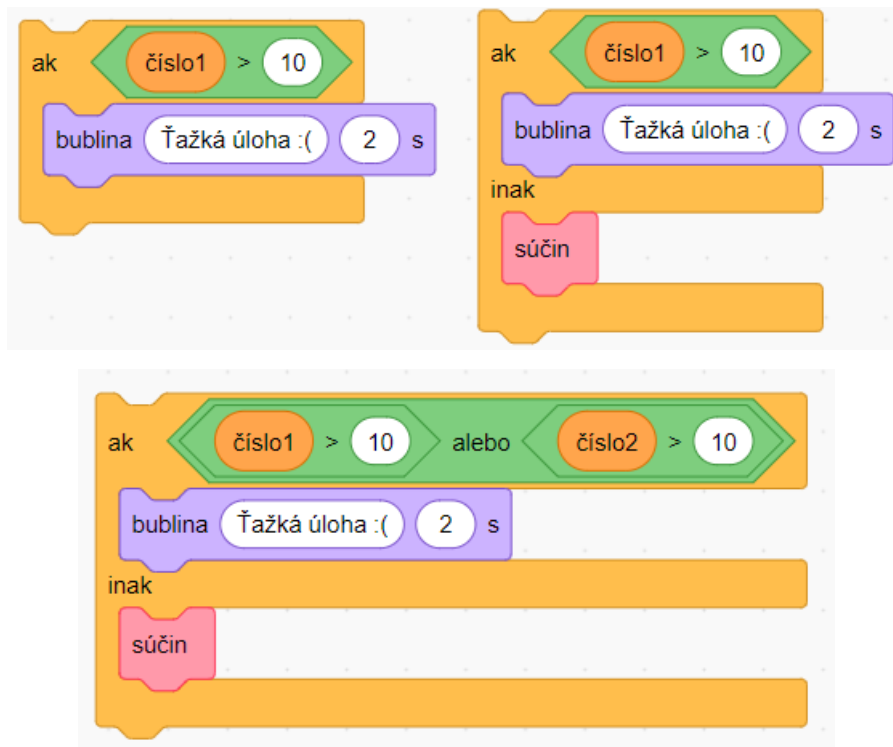
Blok príkazu *ak* má tvar písmena C. Dovnútra sa vkladajú príkazy, ktoré sa majú vykonať podmienene, teda ak je splnená podmienka. Ako podmienka na vykonanie príkazov sa do šesťuholníka vkladá logický výraz.



Blok príkazu *ak-inak* má tvar písmena E. Dovnútra prvej časti sa vkladajú príkazy, ktoré sa majú vykonať, ak je podmienka splnená, do druhej časti sa vkladajú príkazy, ktoré sa majú vykonať, ak je podmienka nesplnená. Tieto dve časti voláme aj **vetvy** podmieneného príkazu.



Vyskúšajte tieto scenáre pre počtárku:




V prvom scenári počtárka len zisťuje, či je *číslo1* väčšie ako 10 a v tom prípade vyhlási, že úloha je pre ňu ťažká. V opačnom prípade neurobí nič.

V druhom scenári v opačnom prípade vypočíta súčin čísel pomocou vlastného príkazu *súčin*.

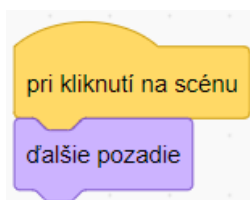
V treťom scenári je podmienený príkaz so zloženou podmienkou, ktorá je pravdivá, ak jedno alebo druhé číslo je väčšie ako 10. Vtedy počtárka vyhlási úlohu za ťažkú, inak úlohu vypočíta.

ÚLOHY

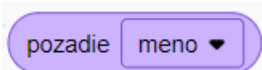
Pokračujeme v projekte Počtárka:

1. Pridajte scéne nové pozadie Chalkboard pomocou ikony . Dokreslite na prázdnu tabuľu text „plus“. Scéna bude mať teraz dve pozadia, jedno s textom „krát“ a jedno s textom „plus“ na tabuli. Pozadia premenujte na *plus* a *krát*.
2. Vytvorte nový blok *súčet*, v ktorom počtárka vypočíta a celou vetou oznámi súčet čísel *číslo1* a *číslo2*. Vyskúšajte počtárku zo sčítovania.
3. Pri kliknutí na scénu meňte pozadie na ďalšie.

Pridaním reakcie scény na udalosť *pri kliknutí na scénu* vieme pomocou príkazu *ďalšie pozadie* cyklicky meniť pozadia scény.



Informácia o tom, ktoré pozadie scény je aktuálne, je uložená vo vlastnosti scény *pozadie*



s parametrom *meno*

4. Vytvorte takýto scenár *pri kliknutí na počtárku*:

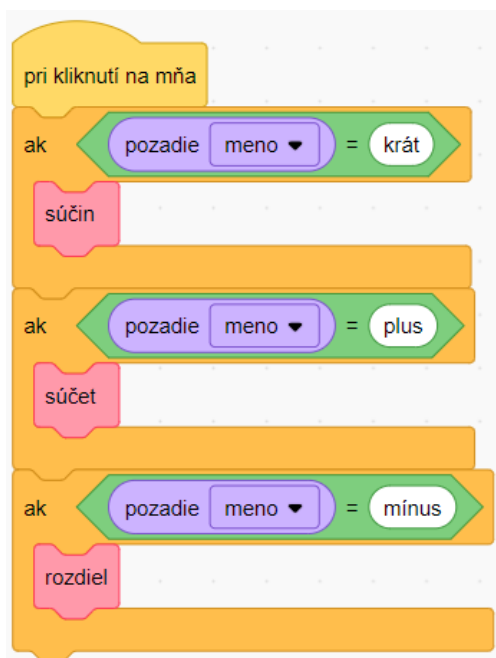
- ak sa meno pozadia rovná *plus*, počtárka oznámi súčet čísel *číslo1* a *číslo2*,
- ak sa meno pozadia rovná *krát*, počtárka oznámi ich súčin.

Použite podmienený príkaz a vlastné nové bloky *súčin* a *súčet*.

ROZŠIRUJÚCE ÚLOHY

Dokončíte projekt Počtárka.

5. Vytvorte nový blok *rozdiel* pre počítanie rozdielu *číslo1* a *číslo2*. Pridajte pozadie s menom *mínus* a textom „mínus“ na tabuli. Pridajte do scenára počtárky počítanie rozdielu. Pomôcka: Pridajte ďalší podmienený príkaz *ak*:

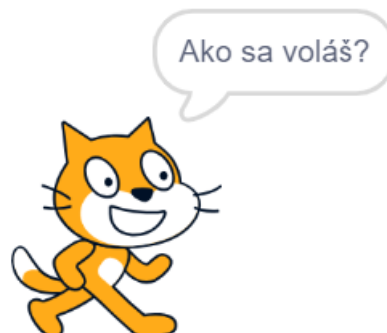
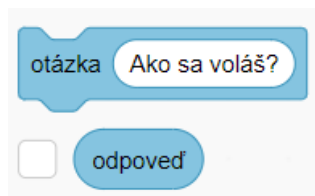


6. Vytvorte nový blok *podiel* pre počítanie podielu *číslo1* a *číslo2*. Pridajte pozadie s menom *delené* a textom „delené“ na tabuli, pridajte do scenára počtárky počítanie podielu. Použite ďalší podmienený príkaz *ak*.

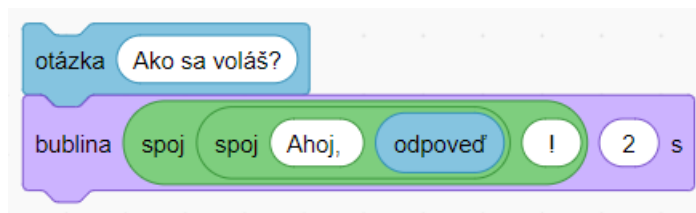
7. Pridajte do príkazu *podiel* ošetrovanie delenia nulou: Ak je *číslo2* rovné 0, počtárka povie, že delenie nulou nemá zmysel, inak vypočíta podiel. Použite podmienený príkaz *ak-inak*.

2.5 VSTUPY A VÝSTUPY

Číselné a textové údaje sa dajú počas behu programu vkladať aj používateľom prostredníctvom **vstupu z klávesnice**. V programovacom prostredí Scratch na to slúži blok s príkazom *otázka*. Postava, ktorá vykonáva príkaz, položí v bubline otázku zo vstupného parametra príkazu *otázka*. Objaví sa vstupný riadok s kurzorom, do ktorého používateľ napíše odpoveď. Odpoveď sa uloží a je k dispozícii prostredníctvom oválneho bloku *odpoveď*.



Údaj získaný zo vstupu od používateľa v odpovedi na otázku sa dá ďalej v programe spracovať. Napríklad postava kocúra vloží meno získané z odpovede na otázku „Ako sa voláš?“ do pozdravu. Príkaz *bublina* je príkazom **výstupu**, ktorým je personalizovaný pozdrav.



V bloku *odpoveď* sa uchováva vždy posledný vstup od používateľa. Ak je otázok viac, posledná odpoveď prepíše hodnotu predchádzajúceho vstupu v bloku *odpoveď*. Preto, ak chceme získať a uchovať viac vstupných údajov, treba ich uložiť do premenných.

Napríklad odpoveď na otázku „Ako sa voláš?“ uložiť do premennej *meno*, v ktorej zostane uchovaná aj po odpovediach na ďalšie otázky.



ÚLOHY

1. Naprogramujte takýto dialóg kocúra s používateľom. Texty zvýraznené fialovou farbou sú odpovede alebo závisia od odpovedí používateľa.

kocúr: Ako sa voláš?

používateľ: *Peter*

kocúr: Ahoj, *Peter*! Koľko máš rokov?

používateľ: *23*

kocúr: Ešte 77 rokov a dožiješ sa stovky :)

Pridajte nejaké ďalšie vlastné otázky a odpovede.

2. Projekt Počtárka upravte tak, aby čísla do počtárskej úlohy zadával používateľ.

LEKCIA 3

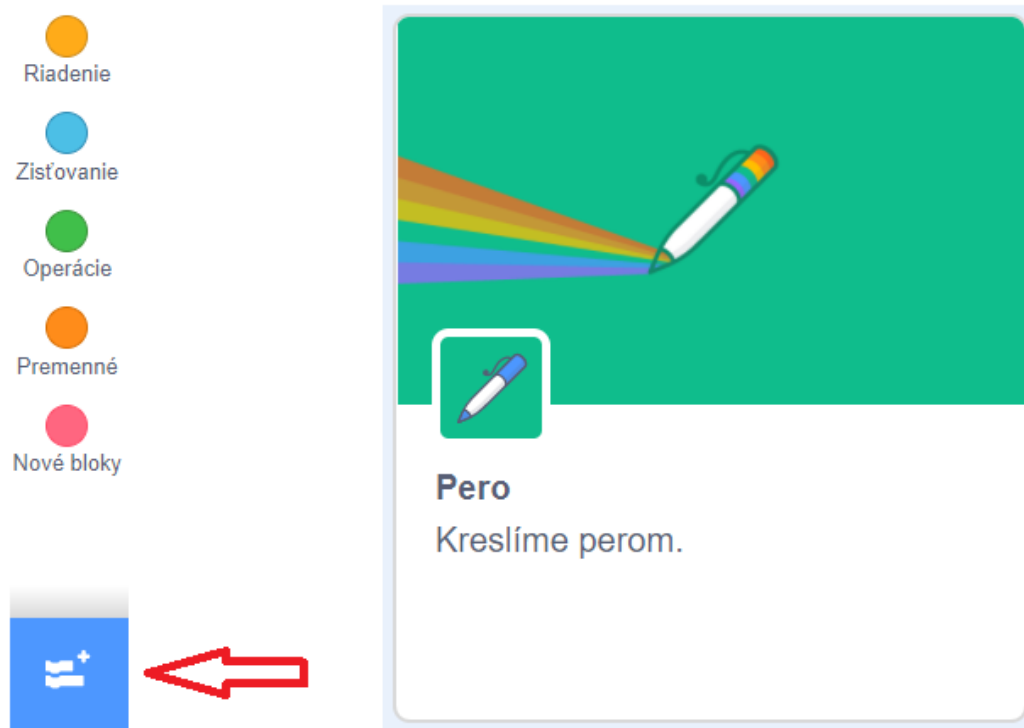
Cieľom tejto lekcie je:

- vedieť rozšíriť prostredie Scratch o nové bloky z rozšírenia,
- používať príkazy z rozšírenia Pero,
- vytvárať obrázky perom rôznych farieb a hrúbok,
- vytvárať obrázky pečiatkovaním kostýmov postavy,
- identifikovať v obrázkoch opakujúce sa vzory, používať cyklus,
- identifikovať v obrázkoch časti vzorov, používať vlastné bloky príkazov,
- definovať vlastný príkaz s parametrami,
- parametrizovať program.

Pracovať budeme s postavou chrobáka ako personifikáciou kresliaceho pera – kresliacim robotom. Na zobrazenie kresliaceho robota sa hodia také kostýmy postavy, ktoré ju znázorňujú v pohľade zhora.

3.1 ROZŠÍRENIE PERO

V prostredí Scratch vieme kresliť zaznamenávaním pohybu postavy – zanechaním stopy. Bloky príkazov na kreslenie nie sú štandardne súčasťou prostredia Scratch, dajú sa pridať **z rozšírenia Pero**.



Bloky z rozšírenia Pero obsahujú príkazy na prácu s kresliacim perom:

- nastavenie vlastností pera – farby a hrúbky pera,
- zapnutie a vypnutie pera.

Okrem nich sa v rozšírení nachádzajú príkazy:

- *opečiatkuj* – opečiatkuje tvar postavy do pozadia scény,
- *zmaž* – zmaže všetky stopy, ktoré postavy nakreslili do pozadia scény.

Všetky príkazy vedú vykonávať postavy, mazať vie aj scéna.

Bloky príkazov na prácu s perom obsahujú ikonu pera, ktorá indikuje, že blok nie je súčasťou základnej sady blokov, ale bol importovaný z rozšírenia Pero.



ÚLOHY

1. Pridajte do prostredia Scratch rozšírenie Pero.

3.2 KRESLENIE, FARBA A HRÚBKA PERA

Kreslíme pohybom postavy **so zapnutým perom**.

Kreslenie pohybom postavy pomocou príkazov *dopredu*, *vľavo*, *vpravo* voláme **korytnačia grafika**. Názov „korytnačia“ grafika pochádza z historicky prvého programovacieho prostredia určeného pre deti **Logo**, v ktorom sa na kreslenie používala postava korytnačky. Kreslenie zmenou súradníc postavy voláme **súradnicová grafika**.

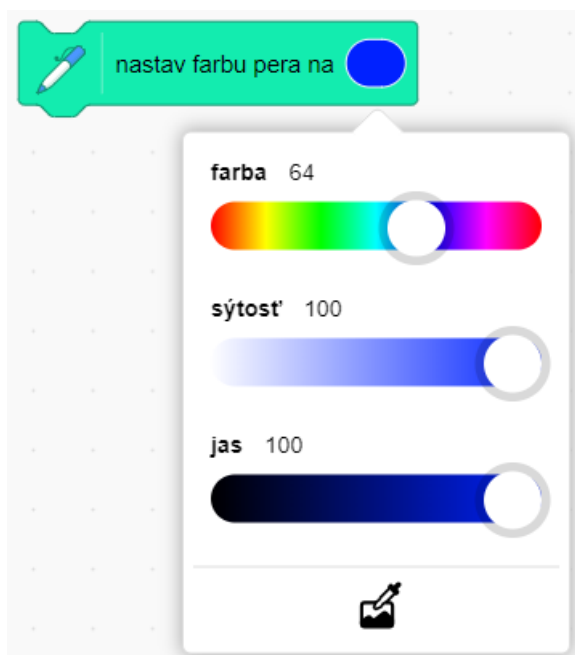
Kreslenie v korytnačej grafike je závislé od pozície a smeru postavy. Kreslenie pomocou zmeny súradníc je absolútne, nezávislé od smeru a pozície postavy.

Nastavme kostým postavy na tvar chrobáka, ktorý bude názorne zobrazovať smer postavy a nebude príliš veľký, aby nezakrýval nakreslený obrázok:

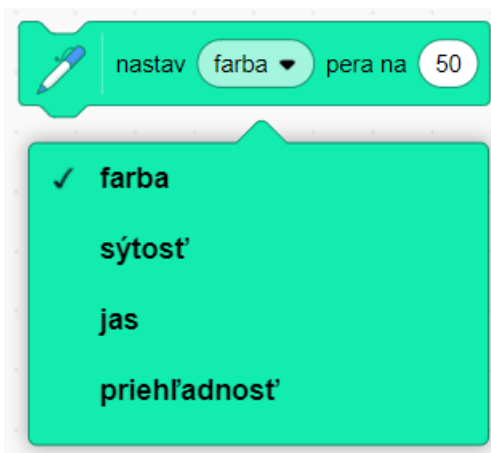


Vyskúšajme pohyb so zapnutým a vypnutým perom. **Farbu pera** nastavujeme príkazom *nastav farbu pera na*. Farbu môžeme zvoliť interaktívne kliknutím na parameter príkazu. Otvorí sa **zmiešavač farby**, v ktorom môžeme nastaviť číslo **farby** v palete farieb dúhy, jej **sýtosť** a **jas**. Nastavujú sa

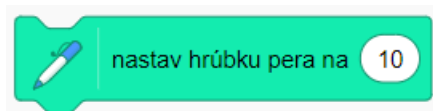
číselné hodnoty z intervalu 0 až 100. V zmiešavači farieb sa nachádza aj nástroj na výber farby interaktívne kliknutím na nejaký bod v okne výstupov.



Vlastnosti farba, sýtosť a jas sa dajú nastavovať aj oddelene a okrem nich aj **priehľadnosť** farby príkazom *nastav* vlastnosť farby *pera na* hodnota. Tento princíp miešania farieb sa nazýva HSV (Hue, Saturation, Value) model.

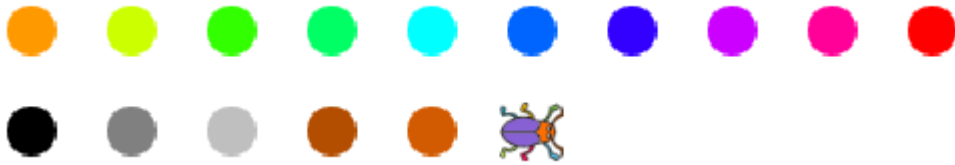


Hrúbku pera nastavujeme príkazom *nastav hrúbku pera na*. Hrúbka pera sa udáva číslom v pixeloch (obrazových bodoch).

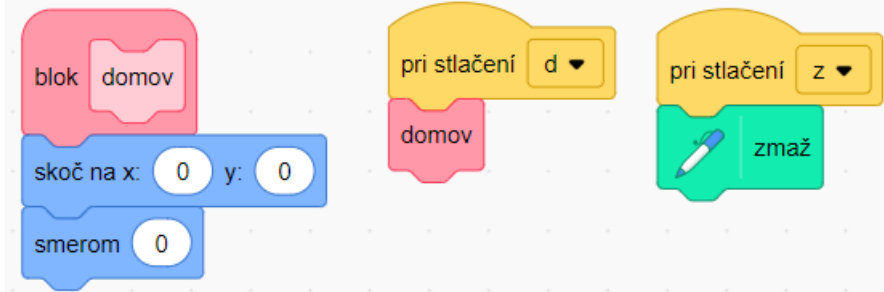


ÚLOHY

1. Vyskúšajte pohyb chrobáka so zapnutým perom s rôznymi farbami a hrúbkami pera, aj s vypnutým perom. Okrem základných farieb dúhy namiešajte aj čiernu, bielu, sivú, hnedú farbu.



2. Naprogramujte pre postavu chrobáka takéto scenáre:



Čo sa vykoná pri stlačení klávesov „d“ a „z“?

3. Naprogramujte vlastné príkazy *strom*, *jablko* a *hruška* a nakreslite takéto obrázky:

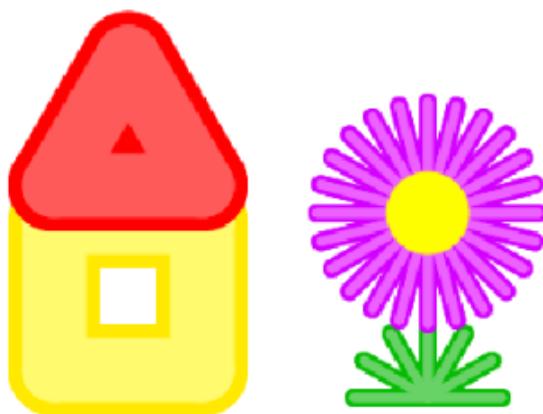


Chrobáka môžete presúvať na príslušné miesto na kreslenie obrázka pomocou myši.

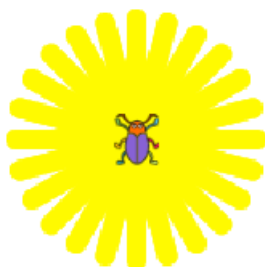
4. S využitím cyklu *opakuj* naprogramujte nové príkazy *kríž*, *kvet*, *štvorec*, *trojuholník*. Používajte rôzne farby a hrúbky pera.



5. S využitím vlastných príkazov *kvet*, *trojuholník*, *štvorec* nakreslite takéto obrázky:



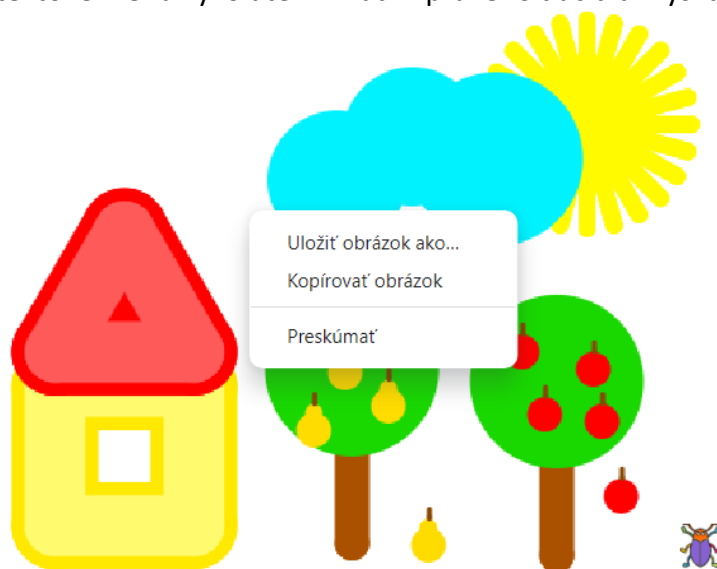
6. Vytvorte nový príkaz, ktorý kreslí žlté *slnko* s lúčmi veľkosti 60. Pridajte scenár, ktorým postava nakreslí na svojej aktuálnej pozícii slnko *pri stlačení klávesu „s“*.



7. Vytvorte nový príkaz, ktorý kreslí svetlomodrý *oblak* zložený z troch prekrývajúcich sa kruhov. Pridajte scenár, ktorým postava nakreslí na svojej aktuálnej pozícii oblak *pri stlačení klávesu „o“*.



8. Nakreslite obraz krajiny z príkazov, ktoré boli riešením predchádzajúcich úloh. Výsledok uložte ako obrázok (kontextové menu vyvoláte kliknutím pravého tlačidla myši do výstupného okna).

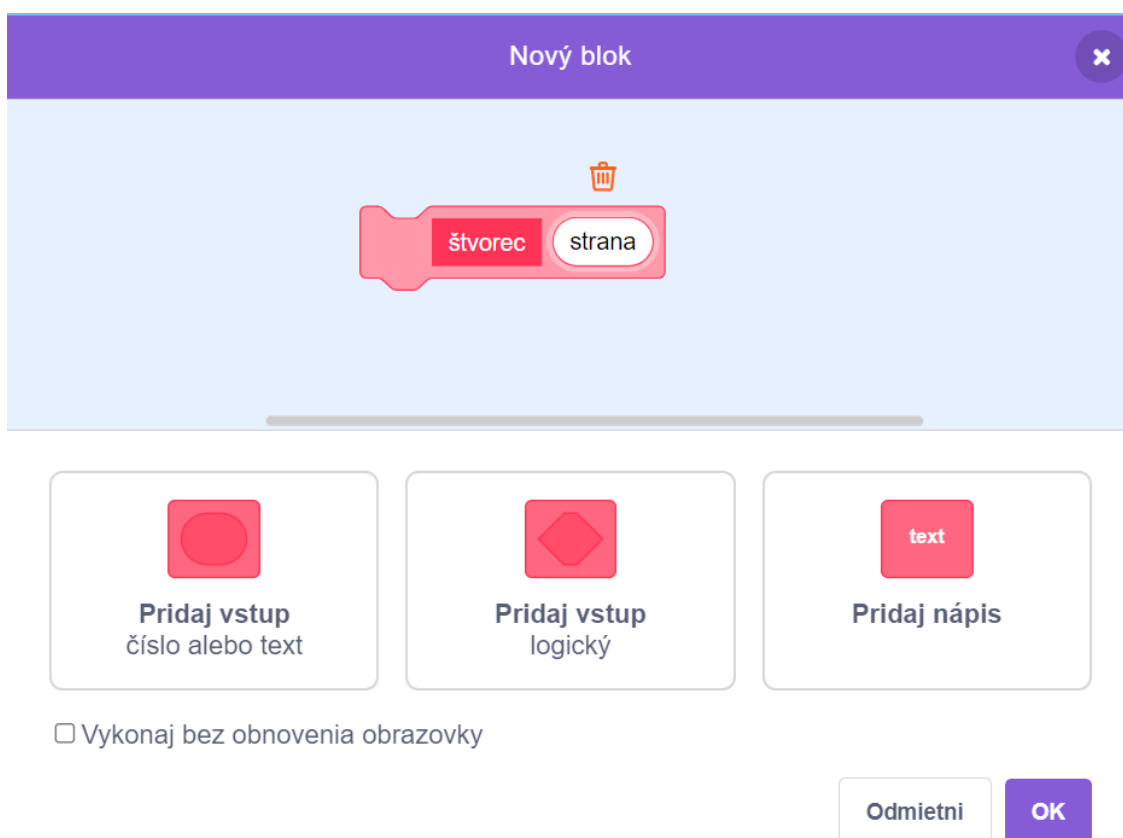


3.3 PRÍKAZY S PARAMETRAMI

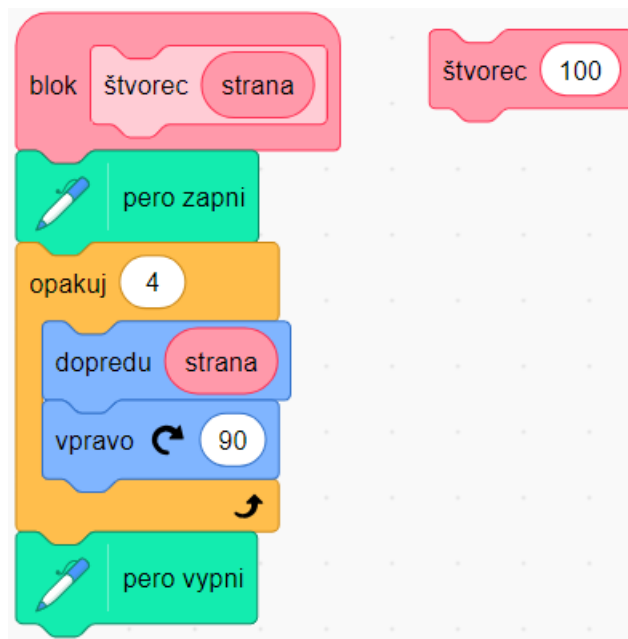
Prostredníctvom **parametrov** vstupujú do príkazov údaje. Príkaz potom negeneruje vždy ten istý výpočet, ale je **hromadný**, to znamená, že generuje rôzne výpočty závislé od vstupných údajov. Napríklad príkazom *dopredu* sa nevykonáva vždy to isté, ale postava sa posúva v aktuálnom smere o vzdialenosť danú hodnotou vstupného parametra.

Aj príkazy naprogramované používateľom (ružové bloky z kategórie Nové bloky) môžu mať jeden alebo aj viac parametrov.

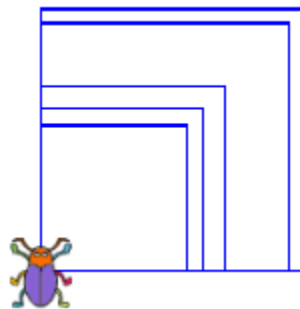
Naprogramujme príkaz *štvorec* s parametrom *strana*, ktorý nakreslí štvorec s veľkosťou strany danou parametrom. Vytvoríme nový príkaz, upravíme jeho meno na *štvorec*, tlačidlom **Pridaj vstup číslo alebo text** pridáme vstup typu číslo a upravíme jeho meno na *strana*.



V editore scenárov naprogramujeme kreslenie štvorca s veľkosťou danou hodnotou vstupného parametra *strana*. Parameter *strana* (blok oválneho tvaru) vložíme do bloku *dopredu* ťahaním myšou z hlavičky bloku *štvorec*. Nový blok *štvorec* má tvar s oválnym číselným vstupom pre zadanie veľkosti strany. Nakreslime štvorce rôznej veľkosti.

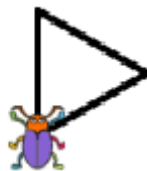


Vyskúšajme niekoľkokrát príkaz *štvorec* na kreslenie štvorca náhodnej veľkosti:

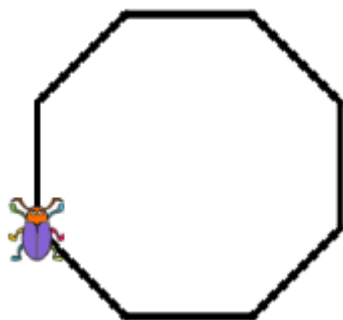


ÚLOHY

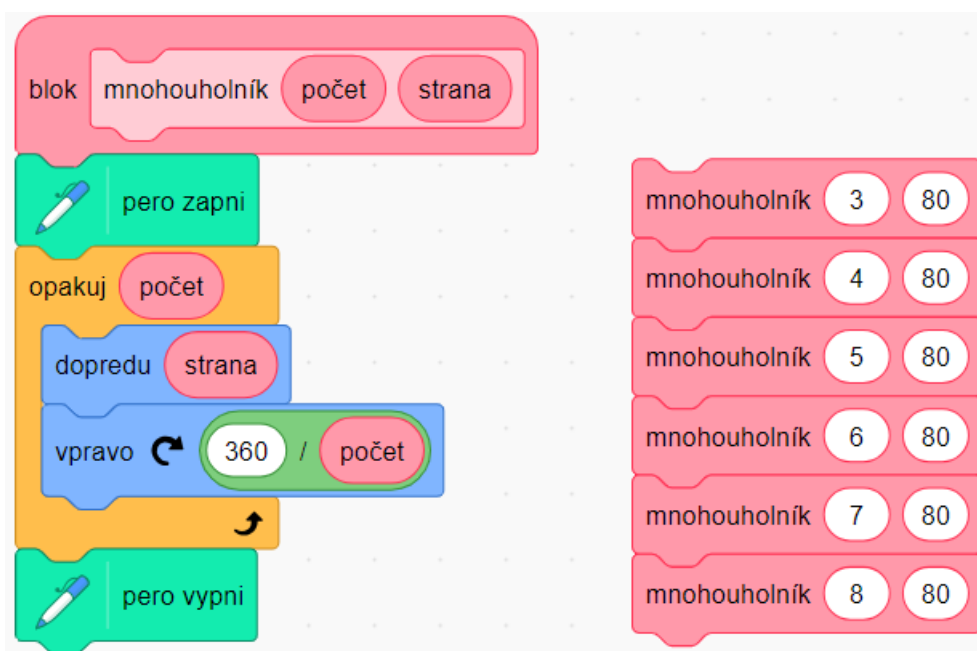
1. Vytvorte príkaz *trojuholník* s parametrom *strana*, ktorý bude kresliť trojuholník s rovnakými stranami danej veľkosti.



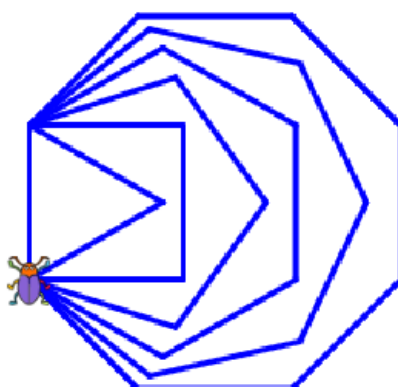
2. Vytvorte príkaz *osemuholník* s parametrom *strana*, ktorý bude kresliť osemuholník s rovnakými stranami danej veľkosti.



Zovšeobecnieme príkazy na kreslenie troj-, štvor-, osemuholníka, a naprogramujeme príkaz *mnohouholník*, ktorý bude kresliť akýkoľvek pravidelný rovnostranný mnohouholník. Okrem parametra *strana* bude ďalším parametrom príkazu *počet* strán mnohouholníka. Vyskúšajme príkaz *mnohouholník* s rôznymi hodnotami parametra *počet*.



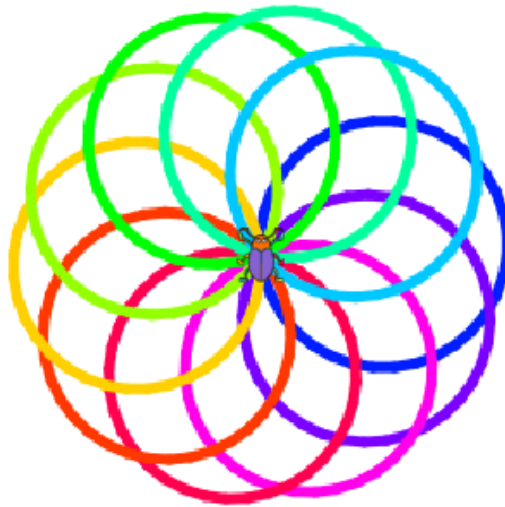
Výsledkom bude takýto obrázok:



ÚLOHY

3. Pomocou príkazu *mnohouholník* nakreslite kružnicu.

4. Nakreslite v cykle 10 kružníc pravidelne otočených cez celých 360° . Príkazom *zmeň farbu pera* o 10 môžete meniť farbu v odtieňoch dúhy.

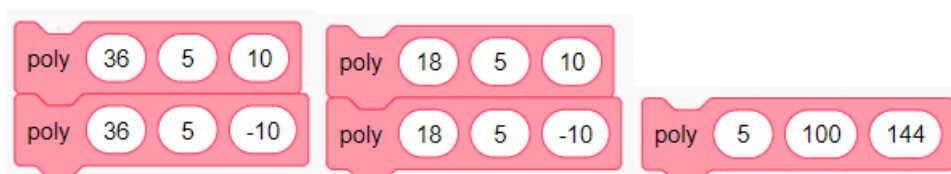


Zovšeobecnieme príkaz *mnohouholník* tak, že aj uhol otáčania sa bude zadávať ako vstup cez tretí parameter *uhol*. Príkaz nazvime *poly* ako skratku z názvu polygón, ktorý znamená mnohouholník.

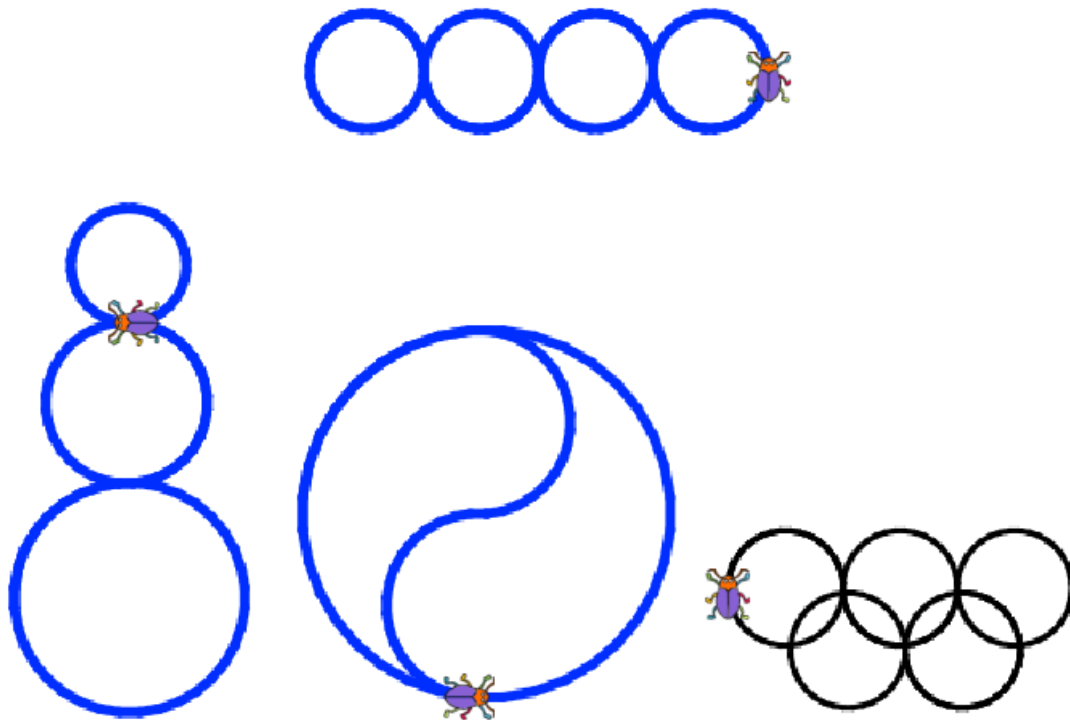


ÚLOHY

5. Čo kreslia nasledujúce scenáre?



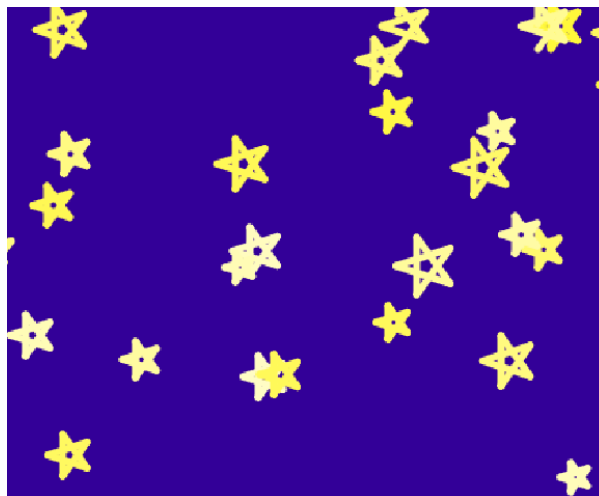
6. Pomocou príkazu *poly* nakreslite takéto obrázky:



7. Vytvorte nový príkaz *hviezda*, ktorý kreslí žltú hviezdu s parametrom *veľkosť*, ktorý udáva veľkosť strany hviezdy. Pridajte scenár, ktorým postava nakreslí na svojej aktuálnej pozícii hviezdu náhodnej veľkosti z intervalu 20 až 50 *pri stlačení klávesu „h“*. Pre pokročilých: Pridajte druhý parameter *odtieň*, ktorým sa dá meniť sýtosť žltej farby. Pri stlačení klávesu „h“ aj ten meňte náhodne.



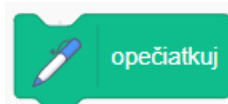
8. Nakreslite hviezdnu nočnú oblohu z hviezd na náhodných pozíciách, náhodného odtieňa žltej a náhodnej veľkosti.



3.4 PEČIATKOVANIE

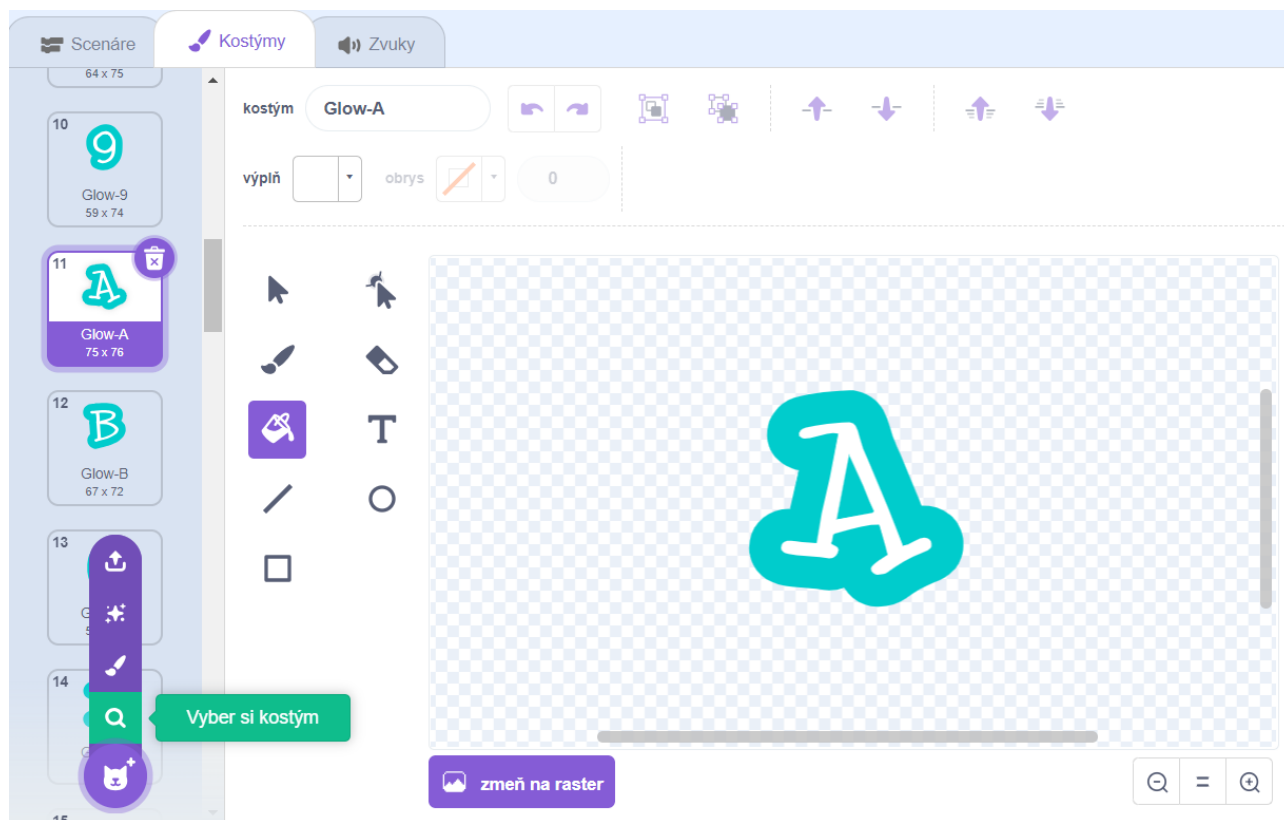
Postava v úlohe kresliaceho pera nie je súčasťou obrázka, ktorý pomocou svojho pohybu so zapnutým perom kreslí, ale ak má kostým, ktorý sa hodí do nakresleného obrázka, môže ho vhodne dopĺňať. Inak je lepšie po nakreslení obrázka postavu skryť.

Okrem kreslenia pomocou pohybu so zapnutým perom vie postava odtlačiť svoj obraz (kostým) na pozadie scény ako **pečiatku**. Postava odtlačí svoj kostým do pozadia scény príkazom *opečiatkuj* z rozšírenia Pero.



Rôzne kostýmy postavy môžu tvoriť kolekciu pečiatok. Príkazom *zmeň kostým na* vieme vybrať konkrétny kostým-pečiatku, príkazom *ďalší kostým* voliť nasledujúci kostým-pečiatku v poradí.

Vytvoríme postavu s kostýmami písmen abecedy. Obrázky vyberme z knižnice kostýmov prostredia Scratch (napr. zo série Glow) v editore na karte Kostýmy. Môžeme pridať aj ďalšie obrázky na vytvorenie sady pečiatok.



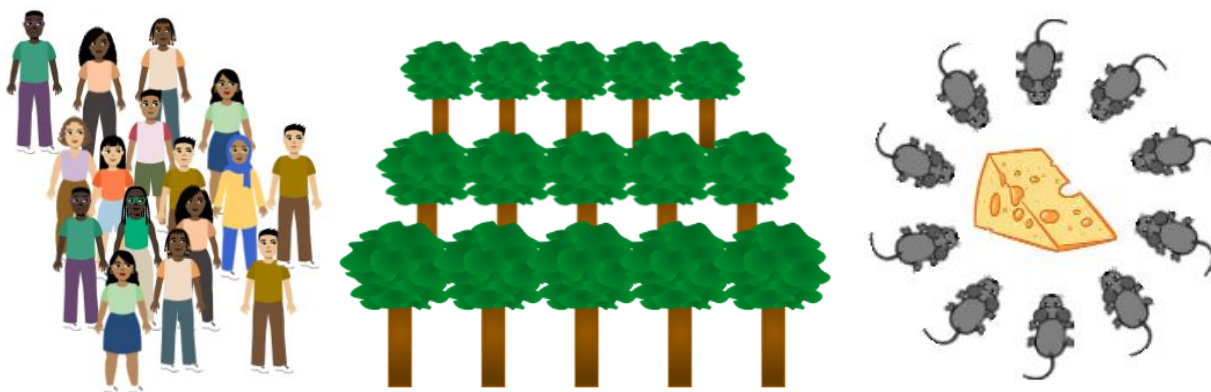
Postavu so zvoleným kostýmom presuňme myšou na vhodné miesto a príkazom *opečiatkuj* vytvoríme jej odtlačok do pozadia scény. Zmenami kostýmu, presúvaním postavy a pečiatkovaním môžeme vytvoriť napríklad takýto obrázok:



Písmená, srdiečko aj kocúr sú rôzne kostýmy tej istej postavy opečiatkované do prázdnej scény.

ÚLOHY

1. Pridajte do projektu ovládanie udalosťami, ktoré uľahčí pečiatkovanie. Napríklad:
 - zmena kostýmu na nasledujúci pri stlačení šípky vpravo,
 - zmena kostýmu na konkrétne písmeno pri stlačení príslušného písmena,
 - opečiatkovanie pri stlačení šípky dolu,
 - presunutie postavy na miesto kliknutia na scénu a podobne.
2. Vytvorte vlastnú sadu pečiatok a pomocou nej napečiatkujte pekný obrázok.



ROZŠIRUJÚCA ÚLOHA

3. Námet na projekt: Pečiatkový editor. Zvoľte si tému a k nej vyberte sadu aspoň piatich obrázkov, ktoré budú slúžiť ako pečiatky na vytváranie rôznych obrazov. Časť scény vyhradte pečiatkam – postavám s rôznymi kostýmami, zvyšok bude plocha na pečiatkovanie. Kliknutím na postavu pečiatky urobte výber pečiatky a kliknutím do plochy na pečiatkovanie odtlačte tvar zvolenej pečiatky. Pečiatkovaním vytvorte pekný obrázok.

LEKCIA 4

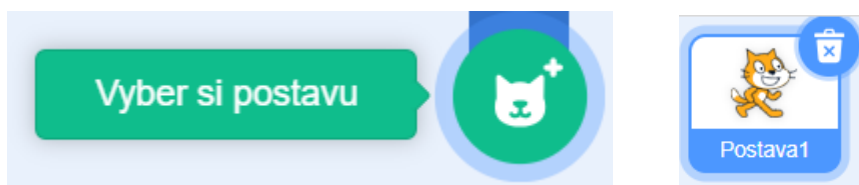
Cieľom tejto lekcie je:

- pridať do projektu viac postáv,
- orientovať sa v štruktúre projektu s viacerými postavami pomocou Prehliadača objektov,
- rozumieť princípu komunikácie objektov v projekte prostredníctvom správ a udalostí,
- tvoriť projekty s viacerými postavami,
- správne riadiť správanie postáv v projekte prostredníctvom udalostí,
- využívať multimedialne prvky – animáciu, zvuky.

Motivačným projektom tejto lekcie je vytvorenie živého obrazu (projekt Farma), v ktorom rôzne postavy animáciou pohybu alebo zvukmi reagujú na udalosti.

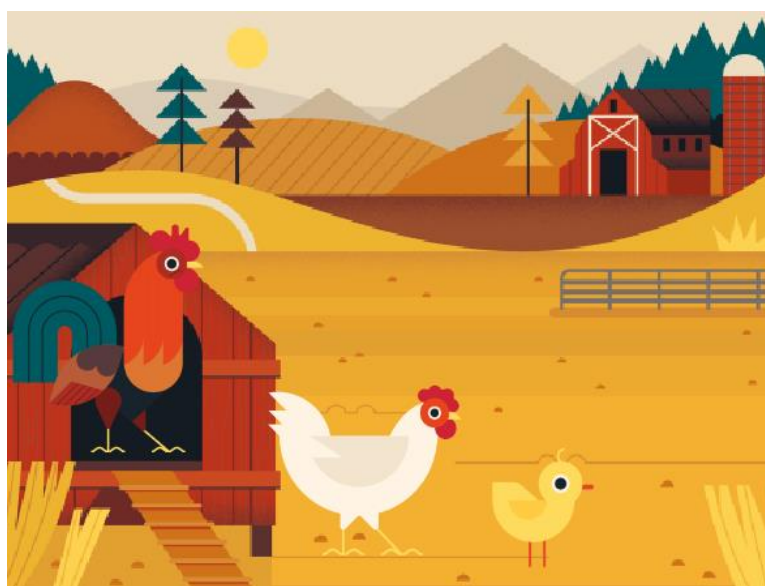
4.1 VIAC POSTÁV V PROJEKTE

V projekte môže byť aj viac ako jedna postava. Do projektu ich vieme pridávať kliknutím na ikonu **Vyber si postavu** v Prehliadači objektov a odstraňovať kliknutím na ikonu **koša** v Prehliadači objektov.

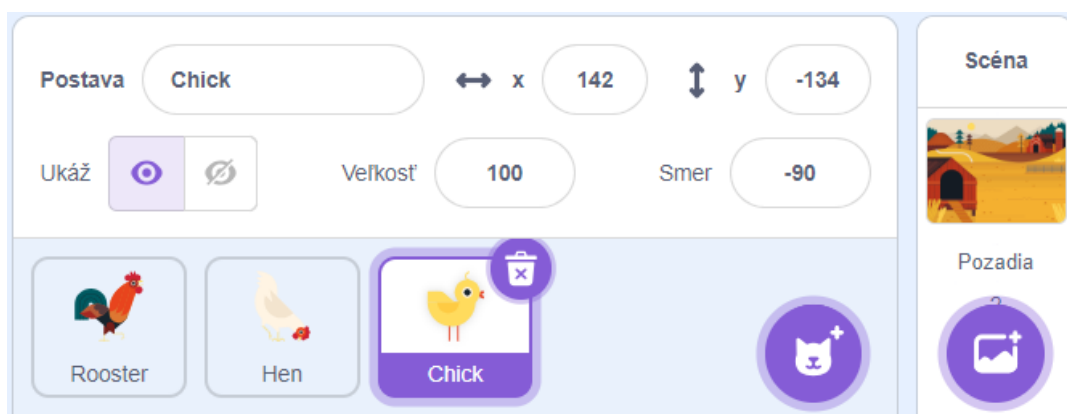


ÚLOHY

1. Vytvorte nový projekt. Pre scénu vyberte pozadie Farm.
2. V Prehliadači objektov odstráňte postavu kocúra.
3. Pridajte na farmu tri nové postavy: kohúta, sliepku a kuriatko.
4. Preskúmajte vlastnosti nových postáv: kostýmy, zvuky.



V prehliadači objektov vidíme tri postavy. Aktuálna postava, ktorej sa týkajú zobrazené vlastnosti v Prehliadači objektov a editory Scenáre, Kostýmy a Zvuky, je zvýraznená.



4.2 ANIMÁCIA A ZVUKY

Vytvorme zo scény s postavami **živý obraz** tak, že postavám naprogramujeme **správanie**, ktoré bude **aktivované udalosťami**.

Naprogramujme kohútovi scenár *pri kliknutí na mňa*: Kohút natiahne krk a zakikiríka prehratím zvuku a zobrazením bubliny s citoslovcom.

Postava kohúta má tri kostýmy: obrázok s hlavou z profilu (rooster-a), obrázok s hlavou z profilu s natiahnutým krkom a nohou (rooster-b), obrázok s hlavou spredu (rooster-c). Zmenou kostýmu príkazom *zmeň kostým na* dosiahneme efekt pohybu. **Animácia** je striedanie obrázkov, ktoré zachytávajú objekt v rôznych fázach pohybu.

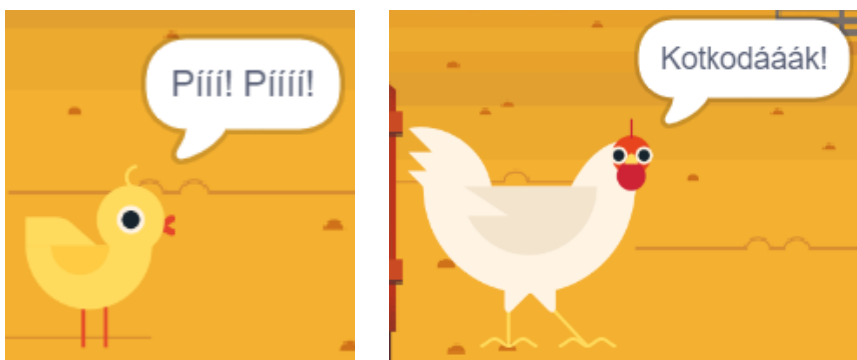
Postava kohúta obsahuje zvuk kikiríkania. K animácii pridáme zvukový efekt prehraním zvuku pomocou príkazu *zahraj zvuk*, a vizuálnou bublinou s citoslovcom kikiríkania. Zvuky vieme editovať, pridávať, mazať na karte Zvuky v editovacej časti vývojového prostredia Scratch.



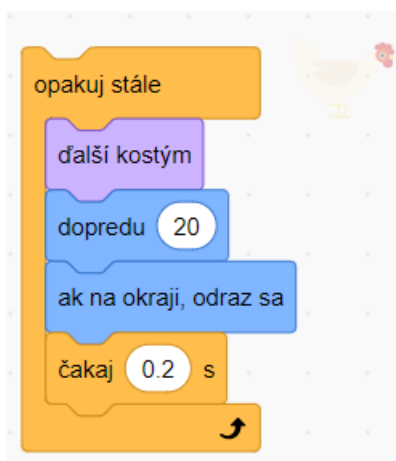
ÚLOHY

1. Analogicky naprogramujte scenáre *pri kliknutí na* kuriatko a sliepku:
 - Kuriatko otvorí zobáček a zapípa prehratím zvuku a zobrazením bubliny s citoslovcom.
 - Sliepka otočí hlavu a zakotkodáka zobrazením bubliny s citoslovcom.




- Nájdiť alebo vytvoriť zvukový súbor s kotkodákaním sliepky a pridať ho ako zvuk postave sliepky.



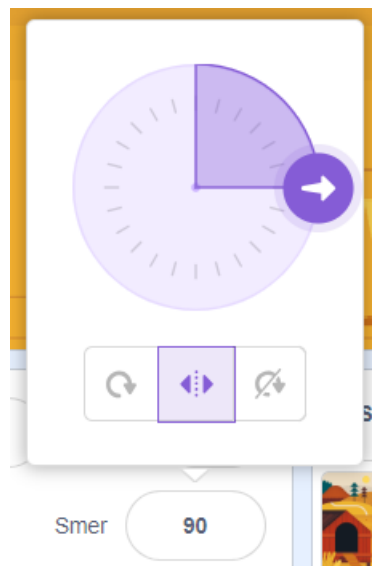
Sliepke naprogramujeme takýto scenár: Sliepka sa bude prechádzať sem a tam donekonečna. V nekonečnom cykle *opakuj stále* posúvame sliepku o kúsok dopredu a zároveň meníme kostým, čím dosiahneme animáciu pohybu. Striedanie obrázkov v animácii spomalíme pomocou príkazu *čakaj*. Príkazom *ak na okraji, odraz sa*, odraz sa otočíme sliepku na okraji scény o 180 stupňov.



Otáčanie postáv sa animuje automaticky v troch režimoch:

-  podľa smeru – postavy menia svoj vzhľad do aktuálneho smeru,
-  vľavo-vpravo – postavy majú len dva vzhľady: jeden pre smery 0 až 180 stupňov, druhý pre smery -1 až -179 stupňov,
-  žiadne – postavy nemenia vzhľad pri zmene smeru.

Režim otáčania sa dá nastaviť v Prehliadači objektov. Kliknutím na vlastnosť *smer* sa zobrazí pomôcka na nastavovanie smeru. Pod ňou sú ikony pre nastavenie troch režimov otáčania, aktívny režim je farebný.



Postavy sa správajú **nezávisle**: Keď nastane nejaká udalosť, zareagujú na ňu príslušným scenárom, alebo nezareagujú, ak scenár pre danú udalosť neexistuje. Na jednu udalosť môže súčasne reagovať viac postáv, teda môže byť definovaných viac reakcií na jednu udalosť, dokonca aj od jednej postavy.

ÚLOHY

2. Animovaný pohyb sliepky sem a tam spustíte ako ďalšiu reakciu na udalosť pri kliknutí.
3. Naprogramujte reakcie sliepky na udalosti *pri stlačení šípky vpravo* a *pri stlačení šípky vľavo* tak, že sliepka nastaví svoj smer na 90 resp. -90 stupňov. Sliepka bude reagovať pri animácii pohybu na stlačenie šípok zmenou smeru pohybu.
4. Všetky zvieratká nech pri stlačení klávesu medzerník naraz predvedú svoju „reč“.

4.3 KOMUNIKÁCIA OBJEKTOV V PROJEKTE

Doteraz sme sa stretli s udalosťami, ktoré generoval používateľ: kliknutie na zástavku, postavu alebo scénu, stlačenie klávesu. Podobne vedú spúšťať scenáre iných objektov aj postavy alebo scéna tým, že **vygenerujú udalosť**, na ktorú iný objekt zareaguje. Udalosti vygenerujú niektoré príkazy postáv alebo scény:

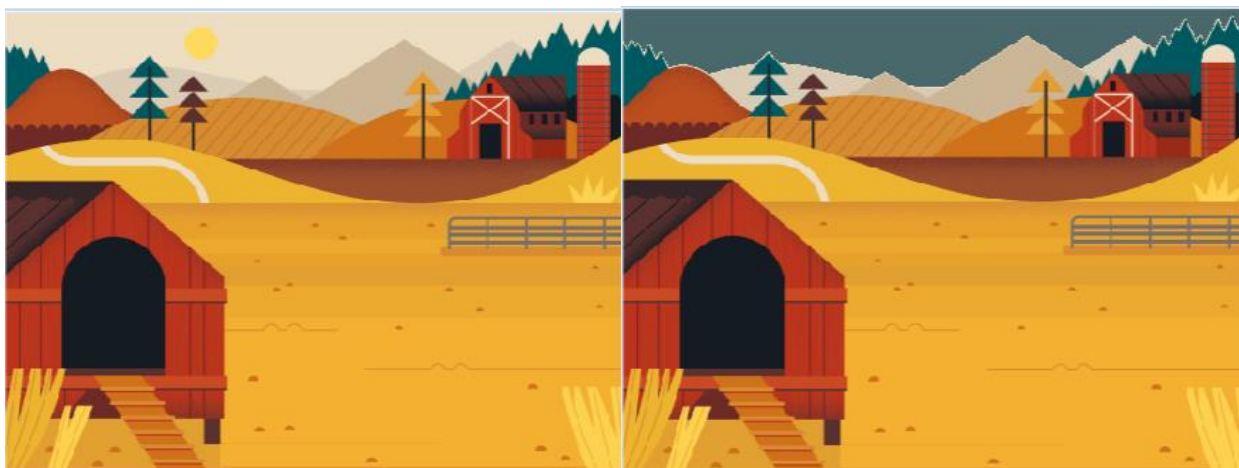
- Scéna automaticky generuje udalosť **zmena pozadia** príkazom *zmeň pozadie*. Ostatné objekty na ňu môžu zareagovať pomocou udalostného bloku *pri zmene pozadia*.
- Scéna aj postavy môžu generovať udalosť **správa** príkazom *vyšli správu*. Ostatné objekty na ňu môžu zareagovať pomocou udalostného bloku *pri prijatí správy*.

Príkazy *zmeň pozadie* a *vyšli správu*, ako aj udalostné bloky *pri zmene pozadia* a *pri prijatí správy* majú parameter, ktorý špecifikuje pozadie a správu.

KOMUNIKÁCIA SCÉNY A POSTÁV PROSTREDNÍCTVOM ZMENY POZADIA

Pokračujeme v projekte Farma. Použijeme zmenu pozadia na ovplyvnenie deja na živom obraze bez interakcie používateľa.

Pre scénu pridajme rovnaké nové pozadie (Farm). Upravme ho graficky na farmu v noci prefarbením oblohy a slnka tmavšou farbou. Názvy pozadí môžeme upraviť na denná farma a nočná farma.



Pri spustení projektu (udalosť *pri kliknutí na zástavku*) spustíme pre scénu scenár, v ktorom sa bude v nekonečnom cykle striedať deň a noc (denné a nočné pozadie scény). Dĺžku dňa a noci nastavíme príkazom *čakaj*.



V cykle sa príkazom *zmeň pozadie* opakovane každých 30 sekúnd mení pozadie a pritom sa generuje udalosť, na ktorú môžu zareagovať ostatné objekty.

ÚLOHY

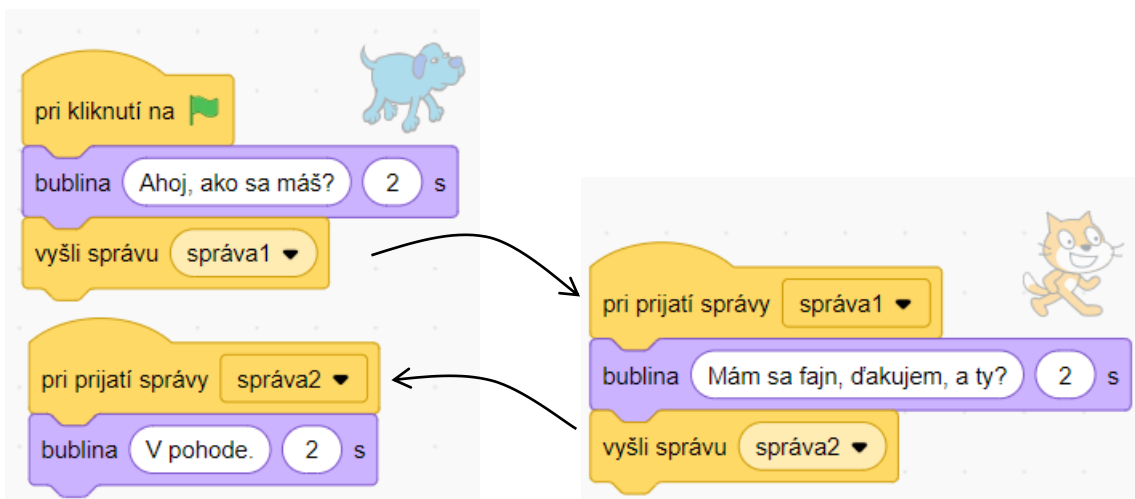
1. Naprogramujte postavám zvieratiek reakcie na udalosť *pri zmene pozadia* takto:
 - pri zmene pozadia na nočnú farmu sa postavy skryjú,
 - pri zmene pozadia na dennú farmu sa postavy ukážu.
2. Kohútovi naprogramujte správanie tak, aby na začiatku dňa, teda *pri zmene pozadia na dennú farmu*, zakikirikal.
3. Pridajte do projektu postavu sovy, ktorá bude mať opačné správanie: ukáže sa *pri zmene pozadia na nočnú farmu*, skryje sa *pri zmene pozadia na dennú farmu*.
4. Pridajte sove správanie: *pri kliknutí* na ňu rozťahne krídla a zahúka zobrazením bubliny. To isté spraví aj bez kliknutia na začiatku noci, teda *pri zmene pozadia na nočnú farmu*.

KOMUNIKÁCIA OBJEKTOV PROSTREDNÍCTVOM SPRÁV

Druhou možnosťou komunikácie medzi akýmikoľvek objektmi v projekte je **vysielanie správ**. Príkazom *vyšli správu* postava alebo scéna vygenerujú udalosť, na ktorú môže reagovať ľubovoľný objekt v projekte scenárom pod udalostným blokom *pri prijatí správy*. Parametrom príkazu a udalostného bloku je správa, ktorú vieme vytvoriť a pomenovať, alebo vybrať z už existujúcich správ.

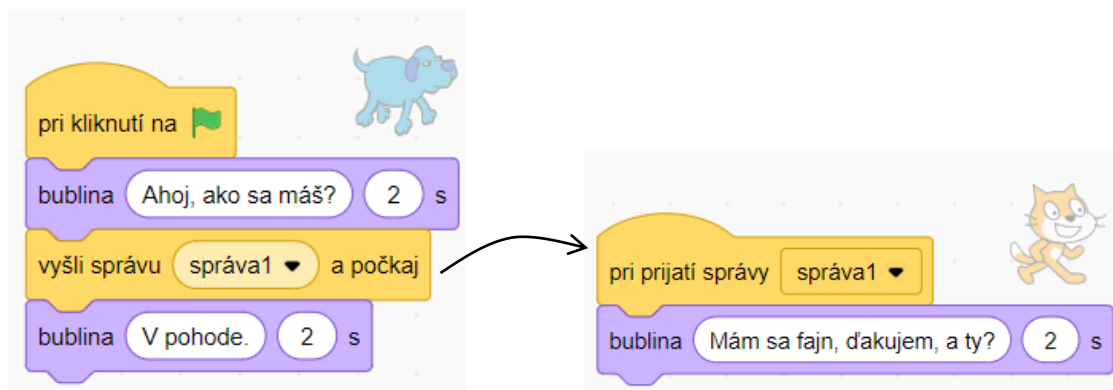


Príkladom komunikácie prostredníctvom správ je jednoduchý rozhovor postáv.



Postava psa začína rozhovor *pri kliknutí na zástavku*. Po otázke vysiela správu *správa1*. Kocúr *pri prijatí správa1* odpovedá a vysiela správu *správa2*. Pes reaguje *pri prijatí správa2* replikou.

V nasledujúcom riešení je použitý iný variant príkazu *vyšli správu*.



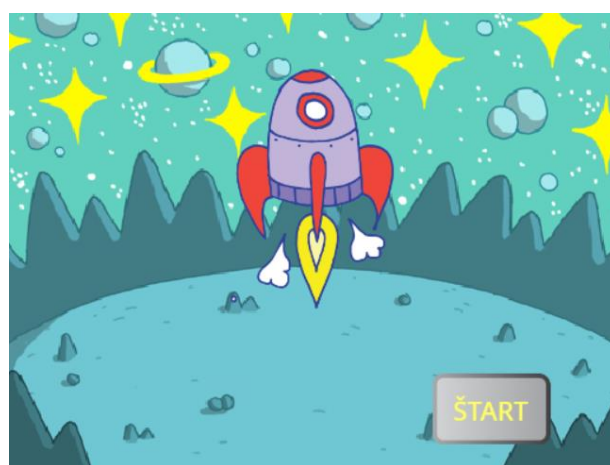
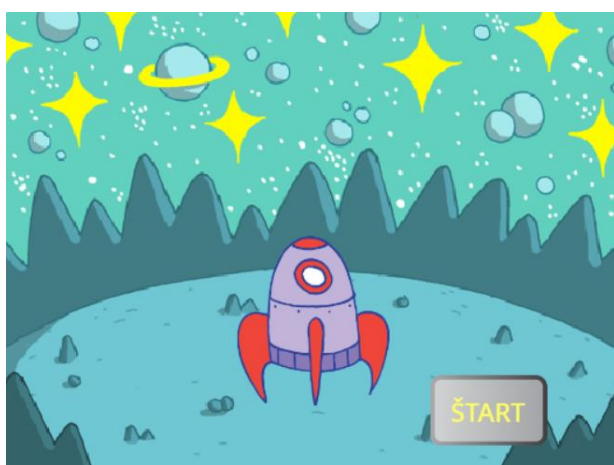
Príkaz *vyšli správu a čakaj* spôsobí pozastavenie vykonávania scenára, kým sa neukončia všetky reakcie na vyslanú správu. Preto nie je potrebné, aby postava kocúra vysiela ďalšiu správu po

dokončení svojej reakcie. Celý rozhovor je riadený scenárom psa, ktorý vysiela správy, čaká na reakciu a pokračuje po reagovaní kocúra.

Ďalším príkladom na komunikáciu objektov v projekte je ovládanie správania postáv prostredníctvom klikania na iné postavy, napríklad ak postava má tvar tlačidla a jeho stláčaním chceme dosiahnuť reakciu inej postavy. Pri kliknutí na tlačidlo sa vyšle správa, na ktorú zareaguje iná postava.

ÚLOHY

1. Pridajte aspoň dve ďalšie repliky do rozhovoru psa a kocúra z príkladu na komunikáciu postáv v projekte.
2. Vytvorte takýto projekt: Na scéne s vesmírnym pozadím je raketa a tlačidlo ŠTART. Pri kliknutí na tlačidlo ŠTART raketa vyletí zo scény smerom hore. Pri kliknutí na zástavku sa obnoví štartovacia pozícia rakety.

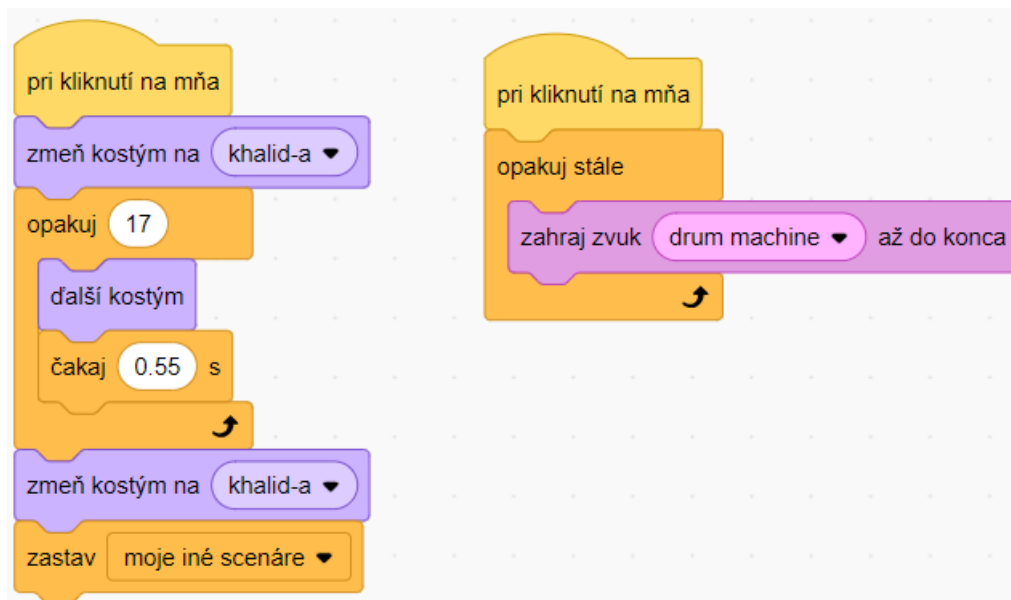


ROZŠIRUJÚCE ÚLOHY

3. Preskúmajte projekt **Tanečný súboj** (<https://scratch.mit.edu/projects/689681283/>). Koľko je v ňom postáv? Aké majú vlastnosti, kostýmy, zvuky?



Preskúmajte scenáre postáv. Na obrázku je scenár postavy Khalid:



Ako sa synchronizuje hudba s animáciou tanca tanečníka? Akú úlohu zohráva príkaz *zastav moje iné scenáre*?

4. Projekt prerobte tak, aby sa v ňom *pri kliknutí na zástavku* odohral takýto **príbeh**:

Postava Khalid osloví svojho súpera a zatancuje svoj tanec. Potom vyzve súpera, aby ukázal svoj tanec. Postava Breakdancer zareaguje na výzvu a predvedie svoj tanec. Spýta sa Khalida, čo povie na jeho tanec. Ten priateľsky prejaví uznanie.

V príbehu sa striedajú aktivity jednej a druhej postavy. Správne poradie a načasovanie riadte prostredníctvom správ.

LEKCIA 5

V tejto lekcii zhrnieme doteraz prezentované koncepty z programovania a systematicky doplníme tie, ktoré sa v ukázkových projektoch z predchádzajúcich lekcí neobjavili.

Cieľom lekcie je:

- mať ucelený prehľad o ponuke príkazov v programovacom prostredí Scratch,
- ovládať algoritmické konštrukcie:
 - cyklus s daným počtom opakovaní, cyklus s podmienkou, nekonečný cyklus,
 - vetvenie úplné a neúplné,
 - podprogram bez parametrov a s parametrami,
- riadiť výpočet udalosťami,
- tvoriť aritmetické, logické, textové výrazy pomocou operácií a zisťovaním stavu prostredia,
- používať jednoduché a štruktúrované premenné,
- vytvoriť postavu a editovať jej vlastnosti počas vývoja v programovacom prostredí a počas behu programu.

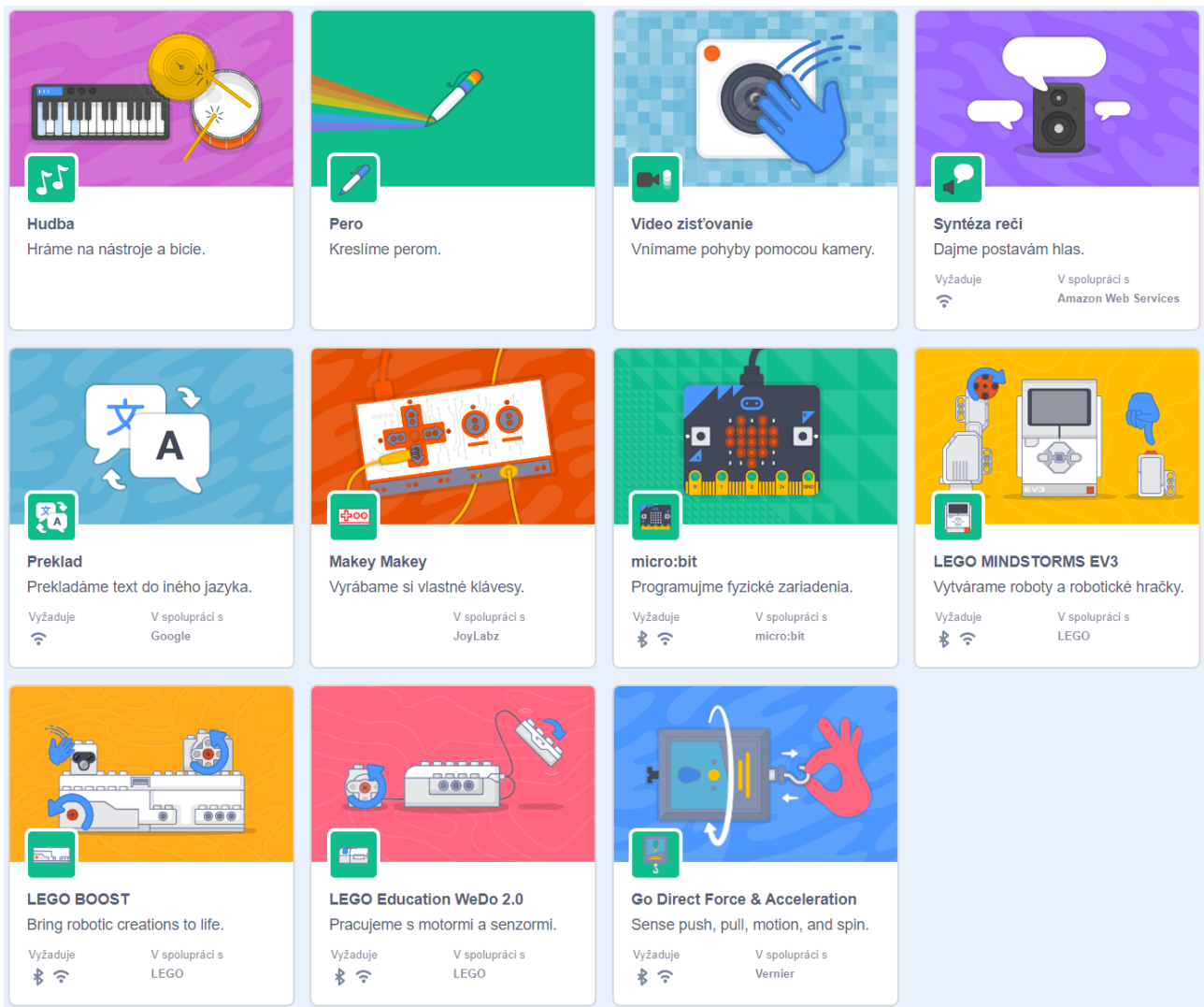
5.1 KATEGÓRIE PRÍKAZOV

Základným motivačným rámcom programovania v prostredí Scratch je tvorba interaktívnych multimediálnych aplikácií. Od toho sa odvíja aj terminológia prvkov programovacieho prostredia (postavy, kostýmy, scéna). Základné príkazy sa preto týkajú pohybu, vzhľadu a zvukov postáv, a vzhľadu a zvukov scény (scéna pochopiteľne nemá príkazy pre pohyb). Príkazy z jednotlivých kategórií sa odlišujú farbou:



Kategórie príkazov je možné rozšíriť pridaním **rozšírenia**. V lekcii 3 bolo predstavené rozšírenie Pero, ktoré obsahuje príkazy na kreslenie pri pohybe postavy. V prostredí Scratch sú k dispozícii ďalšie rozšírenia, ktoré obsahujú ďalšie bloky užitočné pri tvorbe multimediálnych aplikácií alebo bloky na spoluprácu s hardvérom mimo počítača (s mikrokontrolérmi BBC micro:bit, s robotickými stavebnicami Lego, s meracími prístrojmi Vernier).

Prehľad aktuálne dostupných rozšírení s novými kategóriami príkazov je na nasledujúcom obrázku:



5.2 RIADENIE VÝPOČTU

Na riadenie výpočtu slúžia v prostredí Scratch príkazy z kategórií Riadenie a Udalosti. Základné **algoritmické konštrukcie** sú:

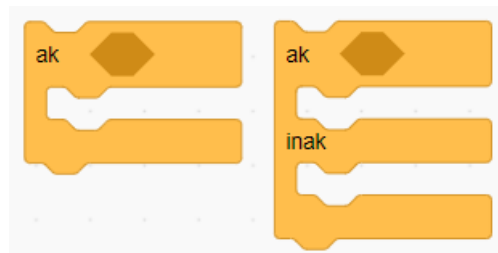
- sekvencia – postupné vykonávanie príkazov,
- cyklus – opakované vykonávanie príkazov
- a vetvenie výpočtu – vykonanie príkazov v prípade splnenia alebo nesplnenia podmienky.

Sekvencia príkazov sa vytvorí jednoduchým spájaním blokov príkazov pod seba do postupnosti.

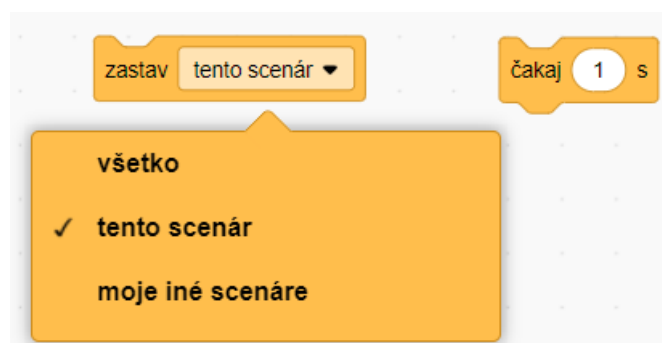
Na opakované vykonávanie príkazov v cykle sú v prostredí Scratch v kategórii Riadenie tri bloky: cyklus *opakuj s* počtom opakovaní ako parametrom, nekonečný cyklus *opakuj stále* a cyklus *opakuj, až kým* s podmienkou ukončenia cyklu ako parametrom, s ktorým sme sa v predchádzajúcich projektoch ešte nestretli:



Na vykonanie príkazov podmienených splnením podmienky slúži príkaz *ak*. Na rozlíšenie dvoch rôznych prípadov pri vykonávaní scenára odlišených splnením resp. nesplnením podmienky sa používa príkaz *ak-inak*:



Okrem týchto algoritmických konštrukcií ponúka prostredie Scratch príkazy na **zastavenie** alebo dočasné **pozastavenie** výpočtu. Príkaz *zastav* má jeden parameter, v ktorom sa zadá, vykonávanie čoho sa má zastaviť (*všetko*, *tento scenár*, *moje iné scenáre*), príkaz *čakaj* pozastaví vykonávanie výpočtu na daný počet sekúnd:

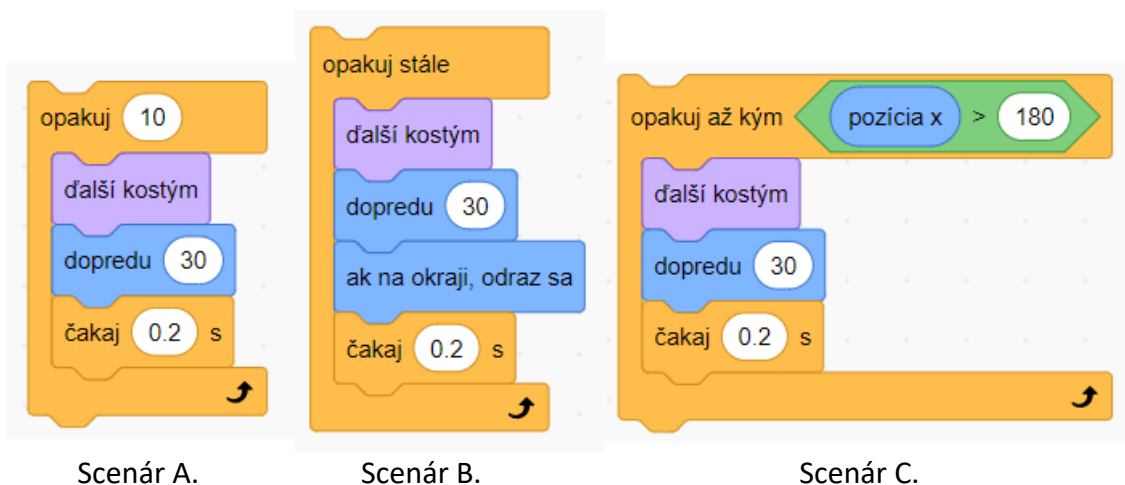


Uvedieme niekoľko príkladov použitia príkazov cyklu, podmienených príkazov, pozastavenia a zastavenia vykonávania scenára.

Vytvorme nový projekt s postavou kráčajúcej Avery (Avery Walking z knižnice postáv), ktorej nastavíme smer na 90 a štýl otáčania *vľavo-vpravo*:



Vyskúšajme nasledujúce scenáre na animáciu pohybu. Zmena fázy pohybu pomocou príkazu *ďalší kostým* a posun *dopredu* sa opakujú niekoľkokrát v cykloch rôzneho typu a niektoré príkazy sa vykonávajú podmienne v podmienených príkazoch. Rýchlosť animácie ovplyvňuje dĺžka pozastavenia vykonávania scenára v príkaze *čakaj*:

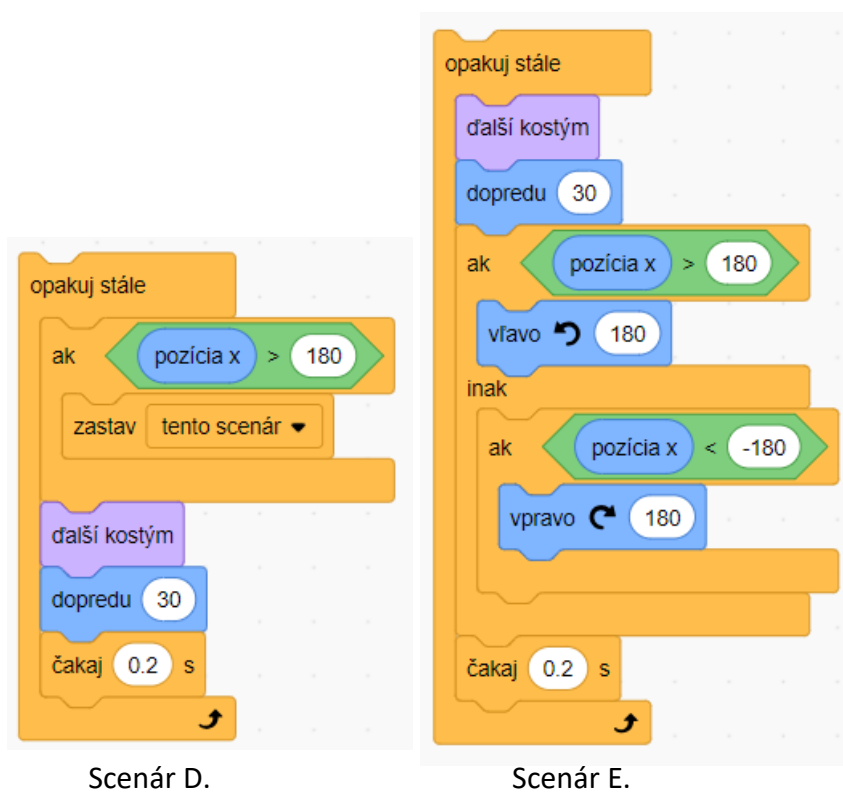


Scenár A.

Scenár B.

Scenár C.

V scenári A sa príkazy opakujú 10-krát (Avery urobí 5 krokov – 10 polkrokov). V scenári B sa príkazy opakujú donekonečna v nekonečnom cykle, preto je do tela cyklu vložený ďalší príkaz *ak na okraji, odraz sa*, aby postava pokračovala v pohybe do opačného smeru, keď narazí na okraj obrazovky. V scenári C je použitý cyklus s podmienkou ukončenia, ktorá je zostavená tak, aby sa pohyb zastavil, keď x-ová súradnica postavy Avery presiahne 180 (dorazí na pravý okraj).



Scenár D.

Scenár E.

V scenári D je ten istý efekt ako v scenári C dosiahnutý použitím nekonečného cyklu, do ktorého je vložený podmienený príkaz *ak* s podmienkou ukončenia a príkazom *zastav tento scenár*. V scenári E sa okrem pravého okraja zisťuje aj dosiahnutie ľavého okraja scény pomocou úplného vetvenia *ak-inak* a ďalšieho vnoreného vetvenia *ak*. V týchto prípadoch postava Avery mení svoj smer na opačný. Efekt je podobný ako v scenári B – postava kráča v nekonečnom cykle tam a späť.

Na **riadenie výpočtu udalosťami** sa používajú bloky z kategórie Udalosti. Udalosti môžu byť:

- vyvolané používateľom: *pri kliknutí* na zelenú zástavku, na postavu alebo scénu, *pri stlačení* klávesu na klávesnici,
- generované programom: *pri zmene pozadia*, *pri prijatí správy*, *pri štarte klonu* (viac v kapitole 5.5),
- vyvolané zmenou stavu prostredia: prekročenie úrovne hlasitosti, uplynutie časového intervalu (viac v kap. 5.4).

V nasledujúcich scenároch reaguje postava Avery na udalosti generované používateľom:



5.3 PODPROGRAMY

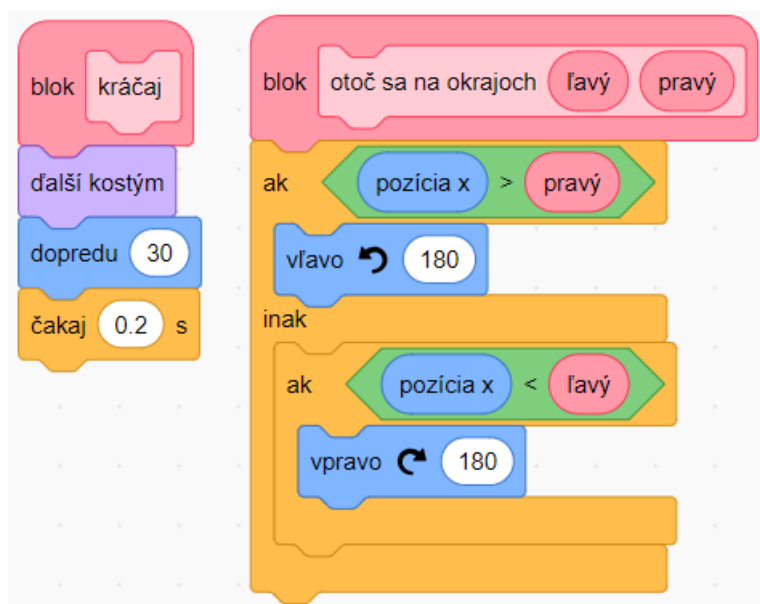
Na štruktúrovanie programu sa používajú podprogramy – v terminológii prostredia Scratch nové bloky. **Podprogram** je pomenovaná časť programu. Pri volaní podprogramu jeho menom sa vykoná programový kód podprogramu a po jeho skončení výpočet pokračuje ďalej za miestom volania podprogramu.

Zmyslom štruktúrovania programu do menších logických celkov je:

- sprehľadnenie programového kódu,
- viacnásobné použitie v rôznych kontextoch.

Možnosti použitia podprogramov v rôznych kontextoch sa zvyšujú, keď má podprogram parametre – vstupné premenné, ktorých hodnoty ovplyvňujú priebeh výpočtu.

Pokračujeme v projekte s kráčajúcou Avery. Uvádzame príklady použitia podprogramov bez a s parametrom. Nový blok *kráčaj* bez parametra animuje pohyb postavy (pol kroku), nový blok *otoč sa na okrajoch* má dva parametre, otáča postavu do opačného smeru, ak je na x-ovej súradnici menšej ako parameter *ľavý* a väčšej ako parameter *pravý*:



Príklady použitia nových blokov (volaní podprogramov) v nasledujúcich scenároch zodpovedajú scenárom A, C, E z predchádzajúcej kapitoly:



Použitie podprogramov zosťručňuje, a tým sprehľadňuje programový kód. Podprogramy sú použiteľné viackrát a parametre umožňujú robiť jednoducho zmeny vo vykonávaní podprogramu.

5.4 PRÁCA S ÚDAJMI

V prostredí Scratch je možné pracovať s údajmi typu:

- číslo – celé a desatinné,
- text (znakový reťazec),
- logické hodnoty – pravda (true) a nepravda (false),
- zoznam (postupnosť) údajov typu číslo alebo text.

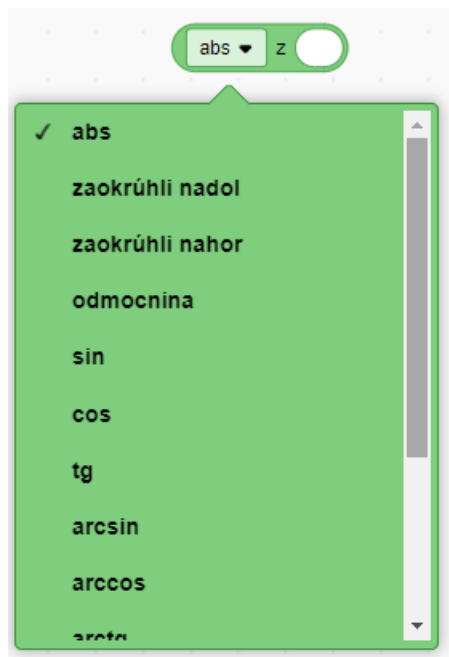
Čísla a texty sú reprezentované oválnymi blokmi, logické hodnoty šesťuholníkmi blokmi. Oválne a šesťuholníkové bloky nie sú samostatné príkazy, ale vkladajú sa do príkazových blokov ako parametre. Číselné a textové údaje môžu byť špecifikované ako **konštanty** – programátor ich vpíše priamo do oválneho bloku, alebo ako **výrazy**.

Výrazy sú oválne alebo šesťuholníkové bloky, ktorých výsledkom je údaj (číslo, text alebo logická hodnota). Väčšinu nájdeme v kategóriách blokov Zisťovanie a Operácie. Do kategórie Operácie patria bloky, ktoré realizujú:

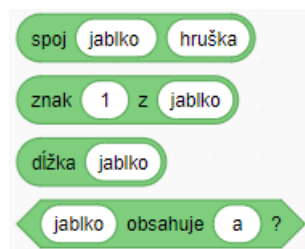
- aritmetické operácie s číslami:



- číselné funkcie:



- operácie s textami:

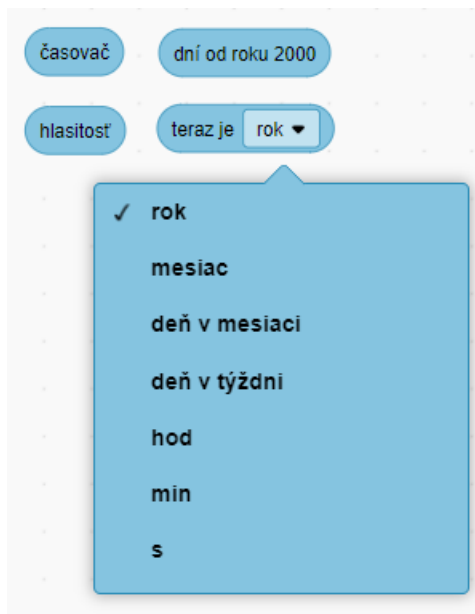


- porovnávacie a logické operácie:

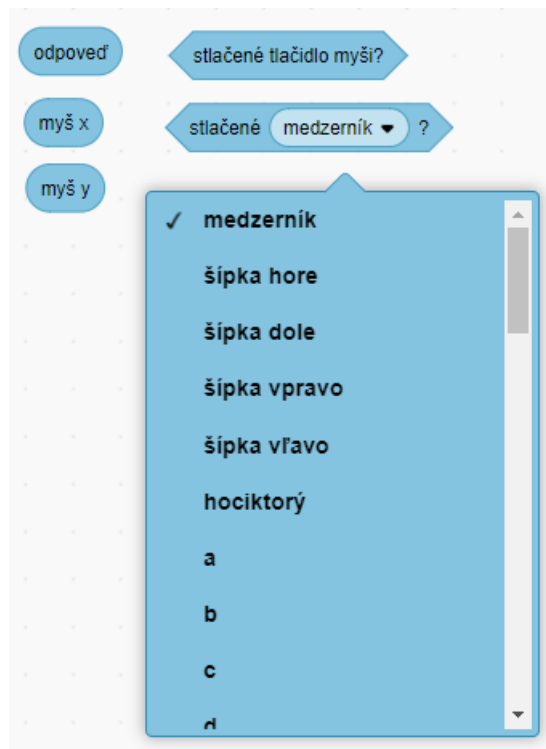


Do kategórie Zisťovanie patria bloky, ktorých výsledkom sú údaje získané počas behu programu:

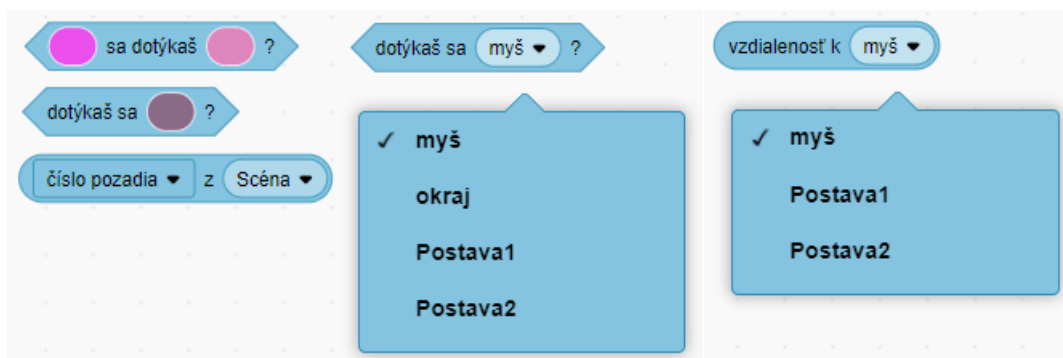
- o parametroch prostredia: čas, dátum, hlasitosti prostredia:



- o vstupoch od používateľa z klávesnice a myši:



- o aktuálnych vlastnostiach a stave objektov v projekte:



Ďalšie bloky na zisťovanie údajov z prostredia je možné importovať z rozšírení.

Ako príklad použitia blokov na zisťovanie stavu objektov v projekte počas behu programu uvidíme ďalší variant prechádzky postavy Avery:

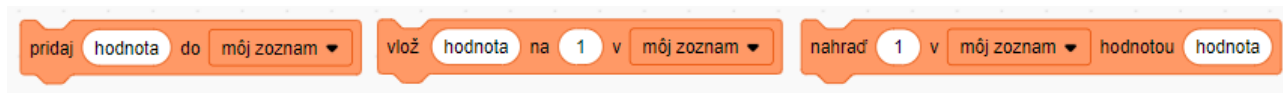


Podmienový príkaz *ak dotýkaš sa okraj vpravo 180* nahrádza príkaz *ak na okraji, odraz sa* v scenári B z kapitoly 5.2. Aký je medzi nimi rozdiel, ponechávame na zistenie čitateľa.

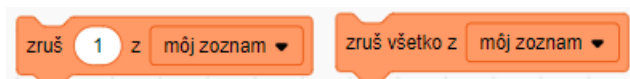
Číselné a textové údaje sa dajú uložiť do premenných. Príkazy na prácu s premennými sa nachádzajú v kategórii **Premenné**. Viac čísel a textov sa dá uložiť do premennej typu zoznam.

Zoznam je **údajová štruktúra** – postupnosť čísel alebo textov, ktoré identifikuje ich poradové číslo v zozname. Na prácu so zoznamami slúžia bloky z kategórie Premenné:

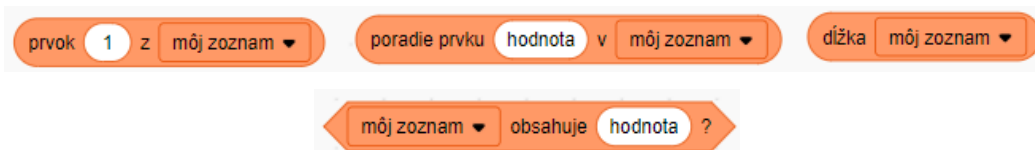
- príkazy na pridávanie prvkov do zoznamu:



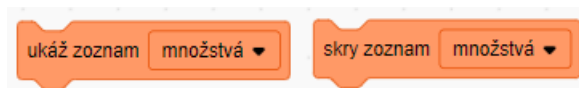
- príkazy na odoberanie prvkov zo zoznamu:



- operácie na zisťovanie vlastností a obsahu zoznamu:



- príkazy na zobrazenie alebo skrytie zoznamu na výstupnej obrazovke:



Údaje v zozname sa dajú editovať aj interaktívne vo vývojovom prostredí pomocou klávesnice a myši:

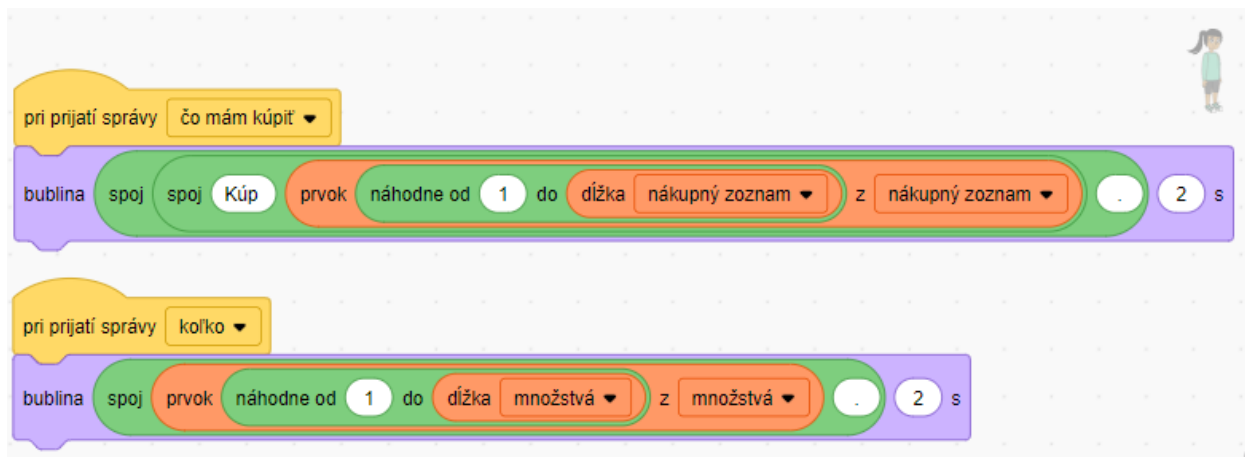
môj zoznam	
1	25
2	Peter
3	1.5 ×
+ dĺžka 3 =	

V nasledujúcom projekte sme vytvorili dva zoznamy – *nákupný zoznam* a *množstvá* ako na obrázku:



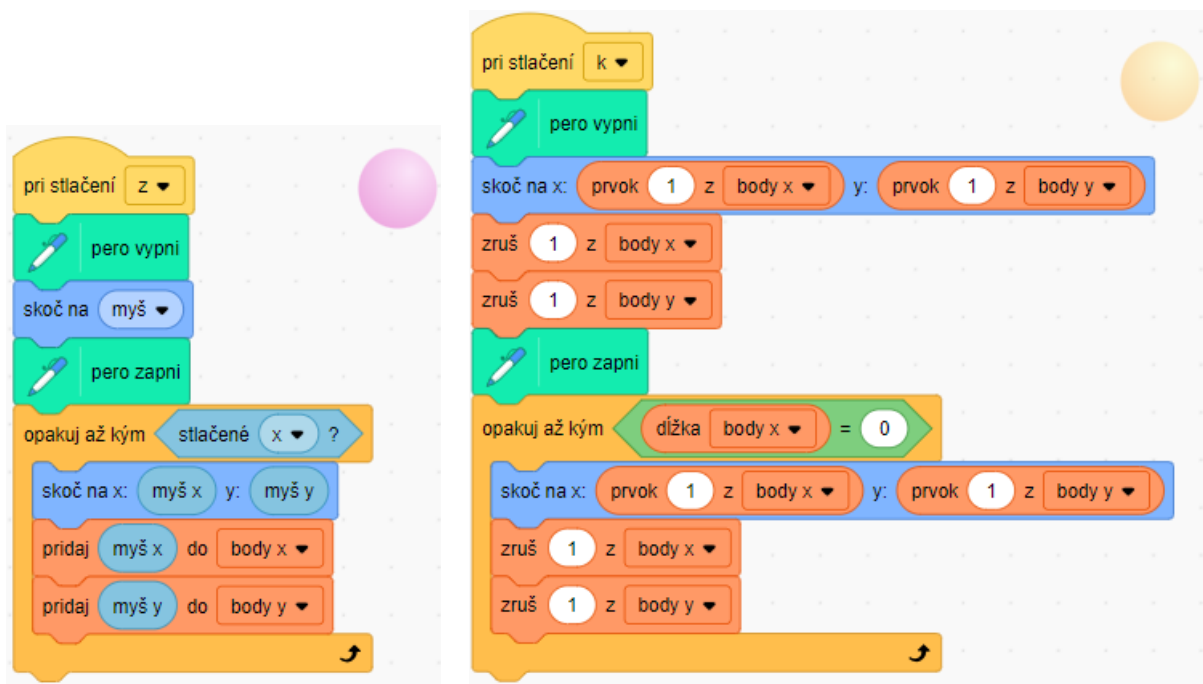
V projekte sú dve postavy, chlapec a dievča, a prebieha medzi nimi jednoduchý dialóg. Chlapec sa spýta, čo má nakúpiť, dievča vyberie náhodne odpoveď zo zoznamu *nákupný zoznam*. Chlapec sa spýta na množstvo, dievča vyberie náhodne odpoveď zo zoznamu *množstvá*. Rozhovor je riadený mechanizmom správ a dievča používa v odpovediach blok *prvok-z* na výber hodnoty zo zoznamu a blok *dĺžka* na určenie dĺžky zoznamu:





V ďalšom príklade demonštrujeme vytváranie zoznamu počas behu programu. V projekte sú dve postavy – ružová a žltá loptička. Ružová loptička na stlačenie klávesu „z“ začne nasledovať pozíciu myši a svoje súradnice zaznamenávať na koniec zoznamov *body x*, *body y* príkazom *pridaj-do*. Pohyb loptičky a ukladanie súradníc do zoznamov sa deje v cykle s podmienkou ukončenia, ktorou je stlačenie klávesu „x“.

Žltá loptička sa na stlačenie klávesu „k“ začne pohybovať po rovnakej predkreslenej dráhe výberom súradníc zo zoznamov *body x*, *body y*. Pomocou príkazu *skoč na* sa premiestni na súradnice na 1. mieste v zoznamoch *body x*, *body y*, a zo zoznamov ich vymaže pomocou príkazu *zruš 1 z body x*, *zruš 1 z body y*. Toto sa opakuje, až kým je zoznam *body x* prázdny (má dĺžku 0).



Na nasledujúcom obrázku vidieť výsledok – pohyb myšou je zaznamenaný ružovou loptičkou tenkým perom a uložením súradníc pozícií myši počas pohybu v zoznamoch *body x*, *body y*. Pohyb žltej loptičky po tej istej dráhe až do momentu vyprázdnenia zoznamu *body x* je znázornený hrubším perom:



5.5 PRÁCA S OBJEKTMI

Pri programovaní v prostredí Scratch sa začínajúci programátor zoznamuje so základnými konceptmi objektového a udalostami riadeného programovania. V projektoch sa môžu vyskytovať objekty dvoch typov (tried) – **postavy** a **scéna**. Výpočet je riadený **udalostami**. Správanie objektov sa naprogramuje zvyčajne ako viac samostatných scenárov, ktorými objekty majú reagovať na vzniknuté udalosti.

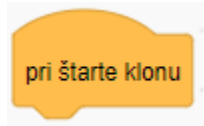
Vlastnosti scény a postáv je možné nastavovať a meniť interaktívne vo vývojovom prostredí alebo príkazmi počas behu programu. Scéna je jedinečný objekt prítomný v každom projekte, ktorý nie je možné odstrániť, ani pridávať viacnásobne. Postavy sú objekty, ktoré je možné pridávať a odstraňovať interaktívne počas vývoja aplikácie vo vývojovom prostredí.

Iba počas behu programu sa dá vytvoriť tzv. „klon“, postavy, ktorý bude mať všetky vlastnosti a správanie pôvodného objektu. Klony nemajú vlastné meno, nedá sa im naprogramovať vlastné správanie, ich správanie je riadené scenármi pôvodnej postavy. Od pôvodnej postavy sa môžu líšiť len hodnotami niektorých vlastností, ktoré sa menia podľa scenára počas behu programu v závislosti od prostredia, v ktorom sa klon vyskytuje.

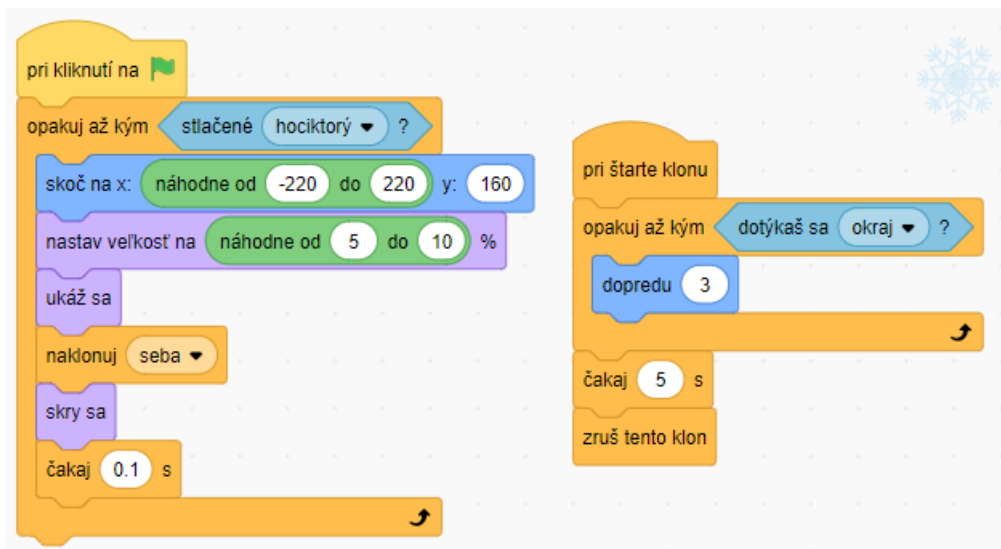
Na vytvorenie a zrušenie klonu počas behu programu slúžia príkazy z kategórie Riadenie:



Pri vytvorení klonu sa vygeneruje udalosť *pri štarte klonu*:



Ako príklad klonovania uvedieme nasledujúci projekt Zima. V projekte má scéna pozadie zasneženej zimnej krajiny s postavou zajaca, ktorý môže alebo nemusí mať naprogramované nejaké správanie. Dôležitejšia je postava vločky, ktorá má aktívne správanie – vytvára svoje klony, ktoré padajú zhora nadol.



Postava vločky má nastavený smer 180, vhodný kostým a veľkosť. Klony vločky sa vytvárajú *pri kliknutí na zástavku* v cykle s podmienkou ukončenia, ktorou je stlačenie ľubovoľného klávesu. Vločka v cykle zmení pozíciu na náhodnú v hornej časti obrazovky, náhodne zmení veľkosť, aby neboli všetky vločky rovnaké, zviditeľní sa, a s takýmito vlastnosťami vytvorí svoj klon pomocou príkazu *naklonuj seba*. Vtedy vznikne udalosť *pri štarte klonu*, na ktorú novo vzniknutý klon zareaguje tak, že sa spustí cyklus, v ktorom naklonovaná vločka padá, až kým sa nebude dotýkať okraja. Po 5 sekundách na spodnom okraji obrazovky klon zanikne príkazom *zruš tento klon* – vločka sa rozpustí.

ZÁVER

V učebnom texte sme predstavili základné koncepty štruktúrovaného a procedurálneho programovania, a jednoduché princípy práce s objektmi a riadenia výpočtu udalosťami. V programovacom prostredí Scratch sa to dá prostredníctvom blokového programovacieho jazyka, ktorý má obmedzené výrazové prostriedky, aby bol jednoduchý na používanie a primeraný aj detskému používateľovi, ale na druhej strane dosť bohatý na to, aby sa pomocou neho dali vytvárať komplexné aplikácie a nevyčerpal sa tvorivý potenciál mladého programátora. Pre takúto vlastnosť edukačného programovacieho prostredia sa používa metafora „**nízky prah a vysoký strop**“. Štruktúrovanú mapu základných pojmov a princíпов programovania v prostredí Scratch uvádzame v Prílohe A.

Vzorové projekty v učebnom texte sú ukážkou typov projektov, ktoré sa dajú v programovacom prostredí Scratch vytvoriť. Väčšinou sme uvádzali ako príklady jednoduché tvorivé prostredia na experimentovanie, krátke multimedialne príbehy alebo živé obrazy. Obľúbeným typom projektov v Scratchi sú aj hry rôzneho typu a simulácie. Zoznam odkazov na vzorové projekty sa nachádza v Prílohe B. Množstvo ďalších projektov od používateľov Scratchu a profesionálnych návodov od autorov prostredia z univerzity MIT, v ktorých možno nájsť inšpiráciu pre vlastný projekt, ponúka programovacie prostredie Scratch na svojom cloudovom úložisku.

Budúcim učiteľom odporúčame na ďalšie štúdium metodické listy pre učiteľov, pracovné listy a učebnice pre deti uvedené v zozname literatúry.

LITERATÚRA

Černochová, Miroslava, Vaňková, Petra a Štípek, Jiří. 2020. *Programování ve Scratch pro pokročilé. Projekty pro 2. stupeň základní školy.* Praha : Univerzita Karlova, Pedagogická fakulta, 2020. ISBN 978-80-7603-085-5.

Kalaš, Ivan a Miková, Karolína. 2020. *Základy programování ve Scratch pro 5. ročník základní školy.* České Budějovice : Jihočeská univerzita v Českých Budějovicích, Pedagogická fakulta, 2020. ISBN 978-80-7394-782-8.

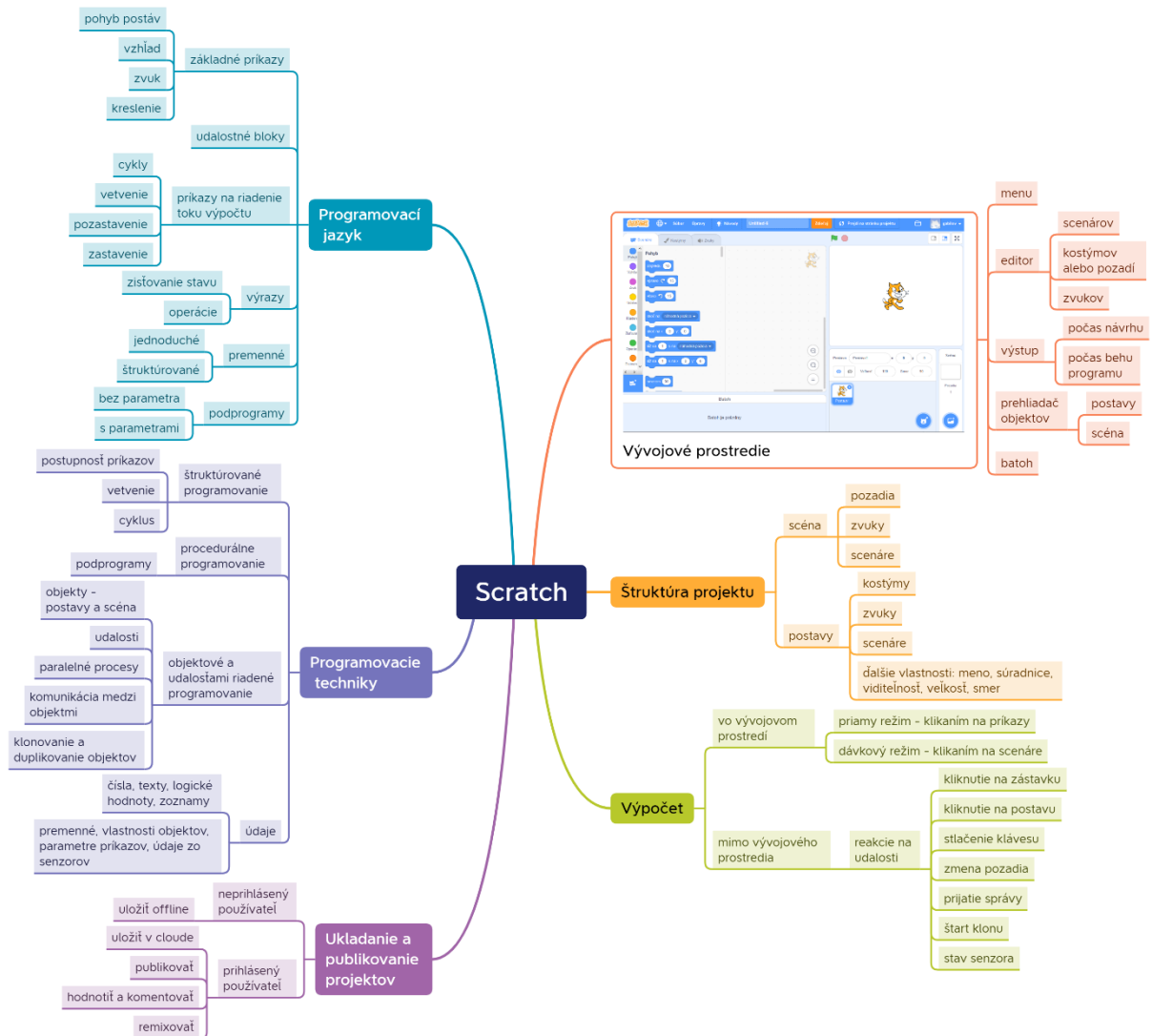
Tomcsányiová, Monika, Hanesz, Angelika a Tkáčová, Zuzana. 2020. *Inovatívne metodiky pre vyučovanie programovania v Scratchi pre základné školy.* Bratislava : Centrum vedecko-technických informácií SR, 2020.

Vaníček, Jiří, Nagyová, Ingrid a Tomcsányiová, Monika. 2020. *Programování ve Scratch pro 2. stupeň základní školy.* České Budějovice : Jihočeská univerzita v Českých Budějovicích, Pedagogická fakulta, 2020. ISBN 978-80-7394-783-5.

Vorderman, Carol. 2022. *Programovanie pre deti.* Bratislava : SLOVART, 2022. ISBN 978-80-556-5643-4.

PRÍLOHY

PRÍLOHA A. KONCEPTUÁLNA MAPA SCRATCH



Presented with xmind

B.1 Mesto – projekt k lekcii 1. Dostupný on-line na: <https://scratch.mit.edu/projects/890684170/>

B.2 Tanečný súboj – projekt k lekcii 4. Dostupný on-line na:
<https://scratch.mit.edu/projects/689681283/>

B.3 Kráčajúca Avery – projekt k lekcii 5. Dostupný on-line na:
<https://scratch.mit.edu/projects/886640097/>

B.4 Nákupný zoznam – projekt k lekcii 5. Dostupný on-line na:
<https://scratch.mit.edu/projects/885820447/>

B.5 Kreslenie rukou – projekt k lekcii 5. Dostupný on-line na:
<https://scratch.mit.edu/projects/889074663/>

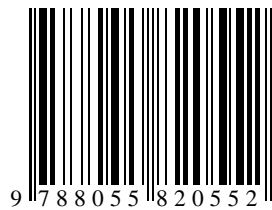
B.6 Zima – projekt k lekcii 5. Dostupný on-line na: <https://scratch.mit.edu/projects/179160392/>

Názov: **Základy algoritmizácie a programovania**
Podnázov: Programovanie v prostredí Scratch
Autor: Gabriela Lovászová

Vydavateľ: Univerzita Konštantína Filozofa v Nitre
Edícia: Prírodovedec č. 821
Formát: A4
Rok vydania: 2023
Miesto vydania: Nitra
Počet strán: 65

ISBN 978-80-558-2055-2

ISBN 978-80-558-2055-2



9 788055 820552