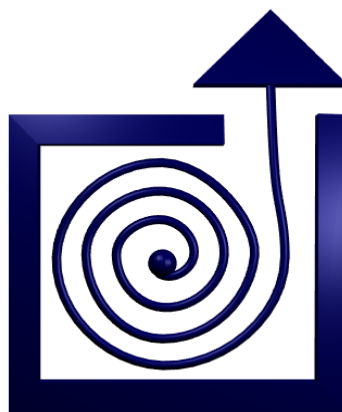


Mgr. Anino BELAN

IMAGINE

učebný text pre terciu osemročného gymnázia



BRATISLAVA
2004

Obsah

Pasieme korytnačku.....	5
Vzdelávanie korytnačky.....	7
Korytnačka s pamäťou.....	9
Korytnačka a parametre.....	11
Rozhodná korytnačka.....	13
Korytnačka s perspektívou.....	15
Vytrvalá korytnačka.....	17
Vnímavá korytnačka.....	20
Vypočítavá korytnačka.....	22
Korytnačka a myš.....	25
Korytnačky.....	28
Korytnačka a rekurzia.....	30
Korytnačka a zoznamy.....	33
Korytnačka mnohých tvárí.....	35
Korytnačka a umenie.....	37
Korytnačka a tlačidlá.....	40
Korytnačka a posúvatka.....	42
Korytnačka a ďalšie rafinované funkcie.....	45
Korytnačka a hodiny.....	49
Korytnačka a potomkovia.....	51
Rozpohybované korytnačky	54
Explodujúce korytnačky	57
Korytnačky a časomiera.....	59
Korytnačka a stránky.....	61
Korytnačka a kalkulačka.....	66

Varovanie pre pedagógov: Napriek tomu, že sa tento text používa v terciách gymnázia Školy pre nadané deti, treba mať na pamäti, že žiaci našej školy majú informatiku od prvého ročníka ZŠ a nie sú úplne neskúsení programátori. Niektoré kapitoly sú aj pre nich pomerne náročné, úroveň vyžadovanej abstrakcie a algoritmickej zručnosti je v nich vysoká a v plnom rozsahu sú ich schopní pochopiť iba niektorí. Texty je pokojne možné použiť aj pri vyučovaní informatiky vo vyšších ročníkoch gymnázia. Pre začiatočníkov, alebo deti nižších vekových kategórií môžu slúžiť ako vítaná inšpirácia, ale vždy treba opatrne zvážiť, či je daný text pre danú skupinu žiakov vhodný. Toto ponechávame na Váš pedagogický cit a majstrovstvo.

Varovanie pre žiakov: Programovanie je niekedy ťažké. Ale má tú výhodu, že ak spravíte hlúposť, následky väčšinou vidíte rýchlo. A ďalšiu, že mašina robí presne to, čo jej povie (aj keď zďaleka nie vždy to, čo by ste chceli). A programovanie (okrem toho, že je ťažké) je aj pekné a ak mu prepadnete, už sa toho nezbavíte.

Anino



Prvá kapitola

Pasieme korytnačku

Dávno pradávno, keď sa voda sypala, piesok lial a počítače skoro ešte ani neboli, vymyslel ktosi programovací jazyk a nazval ho LOGO. Prešlo pár rokov, jazyk sa mal utešene k svetu a kadekto ho používal a tak sa dala dokopy parta ľudí z bratislavského Matfyzu a spravila prostredie, zvané Comenius Logo. I mnohí premnohí sa v tomto prostredí vytešovať mohli, svoje projekty v jazyku LOGO písali a na svete im blažene bolo. Ale čas plynul neúprosne ďalej, procesory zväčšovali svoju rýchlosť aj kapacitu a nereentrantné šestnásť bitové aplikácie zhadzovali operačné systémy a tak sa ďalší vývojári rozhodli, že urobia prostredie nové, Imagine nazvané. A učinili tak. Objektovú stránku jazyka LOGO posilnili, kresliace nástroje povylepšovali, syntax pomenili a bežným užívateľom k dispozícii za rozumnú cenu dali.

Jazyk LOGO má od počítačového praveku až doteraz ústrednú postavu. Je ňou korytnačka v našich zemepisných šírkach a dĺžkach nazývaná Žofka. V závislosti od verzie sa podobá na trojuholník, strelu krátkého doletu, skutočnú korytnačku alebo niečo úplne iné. V prípade Imaginu sa (aspoň zo začiatku) bude na korytnačku celkom aj podobáť. (Pozri obrázok 1.)


Na obrázku okrem korytnačky vidíte aj jednu náramne užitočnú vec: príkazový riadok. (To je ten biely obdĺžnik na spodku.) To je riadok, cez ktorý môžete korytnačke hovoriť, čo má robiť. Zadané príkazy sa objavajú v šedej ploche vyššie. Na obrázku je vidieť posledný zadaný príkaz zmať.

Takže hor sa pilotovať korytnačku. Skúste zadať príkaz **vpravo 90** Korytnačka spraví vpravo-bok a pootočí sa o 90 stupňov. Bude vyzeráť takto:  Podobne po zadaní príkazu **vľavo 60** sa Žofka otočí doľava o 60 stupňov a bude vyzeráť takto:  Vyskúšajte si to.¹



Obrázok 1: Žofka

Úloha č.1: Natočte korytnačku tak, aby po vašich pokusoch zase smerovala hore.

 Keď už viete korytnačku otočiť správnym smerom, môžete ju poslať do sveta. Zadajte jej príkaz **dopredu 100**.² Korytnačka popolezie o 100 korytnačích krokov (korytnačie kroky sú malé, zodpovedajú jednotlivým bodom na obrazovke). A zanechá za sebou stopu, akú vidíte na obrázku. Vláci totiž so sebou atramentové pero a to čarba, kadiaľ korytnačka chodí. Zadajte teraz príkaz **vpravo 90** (korytnačka sa otočí) a potom stlačte šípku hore. V príkazovom riadku sa vám znovu objaví posledný zadaný príkaz. Ak šípku stlačíte ešte raz, objaví sa príkaz, ktorý ste zadali predtým (áno, bolo to **dopredu 100**). Stlačte **<Enter>** a príkaz sa vykoná. Niekoľkokrát za sebou pootočte korytnačku o 90 stupňov a pošlite ju o 100 krokov dopredu. Korytnačka vám nakreslí štvorček a vy sa môžete tešiť. Ak niečo pokazíte, zadajte príkaz **ZNOVU** a celá plocha sa uvedie do pôvodného stavu.

Úloha č. 2: Nakreslite štvorček.

Príkazom **ZNOVU** si zmaťte plochu.

¹ Rada pre lenivších: V tomto prípade fungujú aj skratky. Namiesto **vpravo 90** stačí písať **vp 90**. Podobne sa dá **vľavo 60** skrátiť na **vl 60**. Ak ale píšete príkazy, ktoré má po vás čítať niekto iný, odporúčame používať plné názvy, aby sa v tom dotyčný vyznal.

² Skratka je do 100.

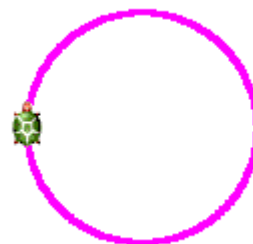
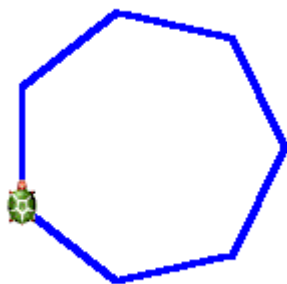
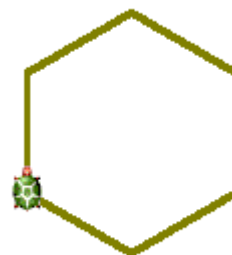
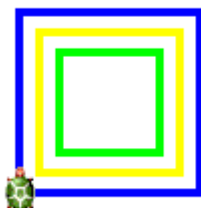
Večné prepisovanie príkazov a preklikávanie sa k nim s pomocou šípiek môže byť po čase únavné a môže liezť na nervy. Preto je užitočné vedieť korytnačke prikázať, aby nejakú činnosť niekoľkokrát zopakovala. Ak napríklad chceme nakresliť štvorec, musíme štyrikrát zopakovať príkazy dopredu 100 vpravo 90. Môžeme to spraviť takto:
opakuj 4 [dopredu 100 vpravo 90]

Za príkaz opakuj napíšeme, koľkokrát to treba zopakovať a do hranatých zátvoriek napíšeme príkazy, ktoré treba zopakovať.

Úloha č. 3: Vyskúšajte, čo spravia príkazy
opakuj 36 [dopredu 100 vpravo 70]
opakuj 180 [dopredu 150 vpravo 178]

Ak chcete, aby korytnačka za sebou čiaru nerobila, zadajte príkaz `peroHore`, ak chcete, aby ju zase začala robiť, zadajte príkaz `peroDolu`.³ Ak chcete vymeniť atrament v pere, napíšete príkaz `nechFarbaPera` "červená (Pozor! Názov farby začína uvozkami, ale nekončí nimi.) Samozrejme môžete zvoliť aj iné farby. Ak chcete zmeniť hrúbku pera, napíšete `nechHrúbkaPera 6` a hrúbka pera bude 6 bodíkov.⁴

Úloha č.4: S pomocou známych príkazov nakreslite nasledujúce obrázky:



³ Skratka pre `peroHore` je `ph` a pre `peroDole` je `pd`

⁴ Skratka pre `nechFarbaPera` je `nechFP` a pre `nechHrúbkaPera` je `nechHP`

Druhá kapitola

Vzdelávanie korytnačky

Ako sa dalo v prvej kapitole zistiť, korytnačka Žofka je veľmi poslušné zviera. Vykoná, čo vám na príkazovom riadku uvidí. V tomto je samozrejme problém. Korytnačka by isto rada vykonala aj to, čo vám na srdci uvidí, ale zatiaľ ešte nebolo vyvinuté dostatočne spoľahlivé rozhranie, ktoré by vedelo prečítať vaše túžby, takže je to na tom príkazovom riadku. A čo si do neho napíšete, to korytnačka (ak tomu porozumie) naozaj urobí.

To, že korytnačka nespraví to, čo chcete, môže mať dve príčiny. Buď porozumie všetkým vašim príkazom a vykoná ich. A vy pozeráte na výsledok a hovoríte si: „Fí ha, toto som naprogramoval ja ?!?!? To som tak teda nemyslel!“ Vtedy treba zadať príkaz **znovu** a zadať príkaz znovu, ale nejak inak, aby korytnačka spravila to, čo chcete. Druhá možnosť je tá, že korytnačke zadáte príkaz, ktorý nepozná. Napríklad chcete, aby korytnačka nakreslila štvorec a tak napíšete **štvorec**. Korytnačka je samozrejme celá zmätená, v živote o takom, že štvorec nepočula, a tak napíše: **Neviem ako sa robí štvorec Buď si sa pomýlil v mene procedúry alebo si ju ešte vôbec nedefinoval** Skrátka, snaží sa ako vládze, v náhlivosti ani čiarky, ani bodky za vetou nepíše, ale nevie. A v tejto kapitole sa dozvieme, ako ju to naučiť.

Je to jednoduché. Ak korytnačku chcete naučiť kresliť štvorce, napíšete jej do príkazového riadku toto:

```
viem Štvorec
dopredu 70 vpravo 90
koniec
```

Akonáhle napíšete príkaz **viem Štvorec**, prepne sa príkazový riadok do učiaceho režimu. Miesto znaku ? Sa na začiatku začne objavovať znak > a žiaden z príkazov, ktorý zadáte sa nevykoná. Všetky príkazy sa ale budú zaznamenávať a pridávať do príkazu **Štvorec**. Tento stav bude trvať až dovtedy, kým nezadáte príkaz **koniec**.

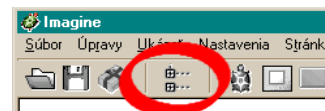
Keď teraz napíšete príkaz **Štvorec**, už sa neobjaví chybová hláška.

Úloha č.1: Naučte korytnačku príkaz **Štvorec** podľa návodu a vyskúšajte ho.

Ak ste postupovali presne podľa návodu, zistili ste, že Žofka príkaz síce vykonala, ale že na štvorec sa to veľmi nepodobá. Dôvod je jednoduchý – uvedené príkazy sme zabudli štyrikrát zopakovať. Túto chybu treba napraviť. Dá sa to spraviť dvoma spôsobmi. Ten komplikovanejší je, že korytnačku naučíme celý príkaz odznova a tentokrát správne. Ten jednoduchší je, že Imaginu siahneme do pamäte a zmeníme príkaz priamo tam.

Najprv si treba zapnúť okno **Pamäť**. Urobíte to tlačidlom vyznačeným na obrázku. Okno Pamäť nám ukáže celé vnútro Imaginu.

Nájdete tu informácie o všetkých objektoch, ktoré sa v ňom nachádzajú, o korytnačkách, udalostiach, plochách a iných veciach o ktorých sme sa zatiaľ nezmiňovali. A samozrejme je tam aj príkaz, ktorý sme práve vytvorili. Keď si teda otvoríte okno Pamäť a rozbalíte chlievik s nápisom **Procedúry**, ukáže sa vám to, čo vidíte na obrázku vľavo. Medzi procedúrami sa vyníma naša nová procedúra **Štvorec**. (Bodaj by sa nevynímala, keď je tam zatiaľ jediná.) Keď na ňu dvojklíknete, otvorí sa vám editor, v ktorom môžete procedúru upraviť.



Úloha č. 2: Upravte procedúru v editore tak, aby skutočne nakreslila štvorec. Vyskúšajte, či to naozaj funguje.

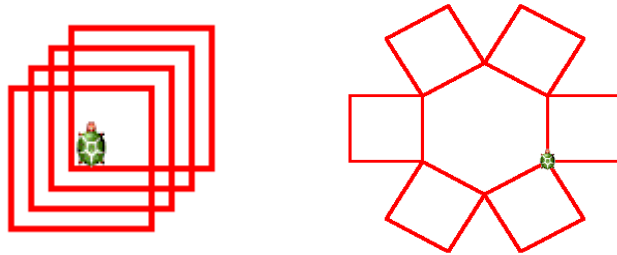
Keď ste upravovali procedúru **Štvorec**, isto ste si všimli, že na okne editora je vľavo dole tlačidlo s nápisom **Pridaj**. S jeho pomocou môžete rovno pridávať nové procedúry a písať ich v editore. Je to pohodlnejšie, ako pridávať procedúry s pomocou príkazu **viem** z príkazového riadku.

Procedúru **Štvorec** teraz môžete používať rovnako ako ktorýkoľvek iný príkaz z **Imagine**. Môže byť dokonca použitý aj v iných zložitejších príkazoch. Vyskúšajte napríklad napísať príkaz

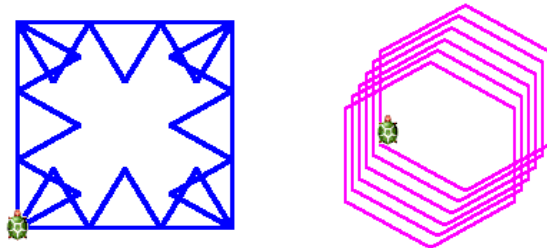
opakuj 24 [Štvorec vpravo 15]

Korytnačka zmúdrela a procedúru vykoná bravúrne, rýchlo a bez problémov.

Úloha č. 3: S pomocou príkazu **Štvorec** nakreslite nasledovné útvary:



Úloha č. 4: Napíšte procedúry **Trojuholník** a **Šesťuholník** a s ich pomocou nakreslite nasledovné útvary:



a vyskúšajte príkaz

opakuj 3 [opakuj 10 [Šesťuholník dopredu 10] vpravo 120]

Úloha č. 5: Napíšte príkaz **Domček**, ktorý nakreslí nasledovný útvar:



Tretia kapitola

Korytnačka s pamäťou

Každá lepšia kalkulačka má aspoň jednu pamäť. Imagine má dokonca okno pamäť.⁵ A ako uvidíte v budúcnosti, aj jednotlivé korytnačky si môžu všeličo pamätať. Kalkulačka si pamätá iba čísla. Pamäť Imagine je všestrannejšia. Môže si pamätať čísla, slová alebo obrázky. A v tejto kapitole sa dozvieme, ako do pamäte niečo vložíť a ako to zas vybrať von.

Predstavte si, že trpíte neprekonateľnou sklerózou a chcete, aby si Imagine niekde zapamätal číslo 5, lebo ak si ho nezapamätá on, vy ho okamžite zabudnete. Musíte si na to číslo vyrobiť krabičku (počítačový guru by povedal „rezervovať pamäť“ alebo „zaviesť premennú“) a tú krabičku nejak pomenovať. Dá sa to spraviť príkazom `urob`. Predpokladajme, že tá naša krabička sa bude volať *číselko*. Ak napíšete príkaz

```
urob "číselko 5
```

udejú sa naraz dve veci. Imagine sa pozrie, či existuje premenná, ktorá sa volá *číselko* a ak nie, tak ju vytvorí. Ak sa pozriete do okna **pamäť**, tak ju tam nájdete v chlieviku s názvom **Globálne premenné**. Druhá vec, ktorá sa udeje je, že Imagine do premennej *číselko* vloží päťku. Môžete sa o tom presvedčiť tak, že dvojkliknete na jej ikonu.



Na uvedenom príklade je jedna nezvyklá vec (aspoň zo slovenčinárskeho hľadiska). Nachádza sa tam totiž len jedna úvodzovka. V Imagine bolo totiž potrebné rozlíšiť názvy príkazov od normálnych slov. Ak to začína úvodzovkou, je to normálne slovo (a príkaz `urob` ho môže použiť ako názov novej premennej). Ak by sme tam tú úvodzovku nedali, Imagine by si myslel, že sa jedná o príkaz *číselko* a sťažoval by sa, že taký nepozná. Ak chcete, môžete si to vyskúšať.

Dobre. S vynaložením istého úsilia sme vyrobili našu prvú premennú. Ako teraz využijť to, čo je v nej uložené? Na to, aby sme sa dostali k hodnote, ktorá je do premennej (po starom do „pamäte“) vložená, potrebujeme ďalší špeciálny úvodzovací znak – dvojbodku. Takže ak napríklad zadáme príkaz

```
píš :číselko
```

Imagine vám vypíše 5. A ak to skúsite s príkazom

```
dopredu 10 * :číselko
```

korytnačka sa posunie dopredu o 50 korytnačích krokov. (Hviezdička znamená „krát“.) Podľa dvojbodky Imagine spozná, že má použiť hodnotu premennej, ktorej meno za dvojbodkou nasleduje.

Do premenných môžeme okrem čísel ukladať aj mnohé iné veci (zoznamy, obrázky, ...) Ak napríklad zadáme príkaz

```
urob "zviera "mamut
```

do premennej *zviera* sa uloží slovo *mamut*.

Ukladanie obrázkov do premenných je troška zložitejšie. Najprv si vytvoríme premennú, v ktorej bude prázdny obrázok. (Prázdny obrázok je taký obrázok, v ktorom nič nie je nakreslené, takže to vlastne skoro ani nie je obrázok. Hlavné je, že sa Imagine k nemu ako k obrázku bude chovať.) Správime to príkazom

```
urob "obrázok prázdnyObrázok
```

⁵ Na rozdiel od Imagine má človek buď pamäť, alebo okno. Keď majú okno Windows, volá sa to BSOD (Blue screen of death).

V okne pamäť nám pribudne premenná `obrázoček`, ale zatiaľ v nej nič nie je. Keď na ňu dvojkliknete, otvorí sa okno so všetkými globálnymi premennými a uvidíte, že rámček vedľa nej je prázdny. A teraz nastala vaša chvíľa. Dvojkliknite do toho prázdneho priestoru. Objaví sa vám program **LogoMotion**, ktorý je určený na kreslenie. V kolónke **Súbor** si vyberte **Nový** (Alebo stlačte **Ctrl-N**) a môžete kresliť. Keď ste hotoví, zavrite LogoMotion a na otázku, či váš výtvor do premennej uložiť odpovedzte **Áno**.

To, čo ste nakreslili, sa stalo obsahom premennej `obrázoček`. Túto premennú môžete používať v príkazoch, ktoré s obrázkami vedia pracovať. Môžete napríklad zmeniť tvar korytnačky príkazom

```
nechTvar :obrázoček
```

Úloha č.1: Zmeňte tvar korytnačky opísaným spôsobom.

Nasledujúce úlohy sú zamerané na manipuláciu s premennými.

Úloha č.2: Aká hodnota bude uložená v premennej `hausnumero` po každom z uvedených príkazov? (Najprv si rozmyslite odpoveď, až potom skúšajte.)

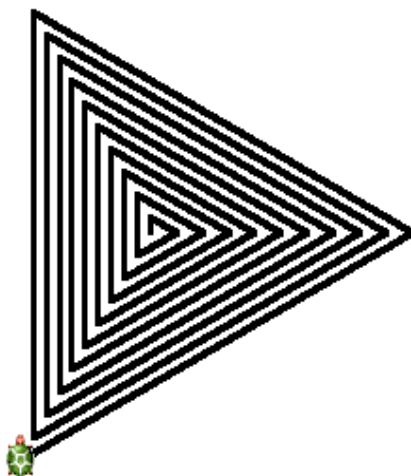
```
urob "hausnumero 3
urob "hausnumero :hausnumero + 4
opakuj 10 [ urob "hausnumero :hausnumero + 2 ]
urob "hausnumero :hausnumero / 9
píš :hausnumero
```

Úloha č.3: Čo spravia nasledujúce príkazy? (Znovu si najskôr premyslite odpoveď.) Aká hodnota bude na konci v premennej `a`?

```
urob "a 1
opakuj 400 [ dopredu :a vpravo 90 urob "a :a + 1 ]
```

Úloha č.4: Vytvorte premenné `hihi` a `hoho`, do oboch vložte číslo 1. Potom do `hihi` vložte súčet čísel, ktoré sa práve v týchto premenných nachádzajú a hneď nato vložte do `hoho` súčin čísel v oboch premenných. Toto vkladanie zopakujte päť krát. Aké hodnoty budú v oboch premenných?

Úloha č.5: S použitím premenných a finty z úlohy č. 3 skúste nakresliť nasledujúci obrázok:



Štvrtá kapitola

Korytnačka a parametre

Už viete, ako korytnačku naučiť nový príkaz. Rovnako si viete vytvoriť a použiť premennú – krabičku v pamäti, do ktorej môžete ukladať čo potrebujete. V tejto kapitole sa dozviete, ako procedúry a premenné súvisia.

Keď sme začali s vytváraním procedúr, možno niekto závistlivo pokukoval po systémových procedúrach – takému príkazu dopredu je sveta žiť. Keď napíšeme dopredu 50, korytnačka ide dopredu o 50 bodíkov, keď napíšeme dopredu 140, ide dopredu o 140. Naše procedúry vykonali čo mali a vždy to vyzeralo rovnako. Zato procedúra dopredu si počká, aké číslo za ňu zapíšete a zachová sa podľa toho. A čo viac – ak napíšete iba dopredu a stlačíte <Enter>, tak sa... čo sa vlastne stane, si vyskúšajte sami.

Taká vec, ako procedúra s parametrom⁶ je veľmi praktická. Nemuseli by sme robiť veľa rôznych procedúr na kreslenie štvorcov rôznych veľkostí. Stačila by jedna. Parametrom by sme jej povedali, aký veľký štvorec potrebujeme a korytnačka by nám taký nakreslila.

No a v tomto momente prichádzajú k slovu premenné. Nebudú to globálne premenné na ktoré by sme sa mohli pozrieť v okne pamäť. Budú to **lokálne premenné**, ktoré budú viditeľné len pre procedúru a do ktorých si bude ukladať vstupné hodnoty. Univerzálnu procedúru štvorec urobíme takto:

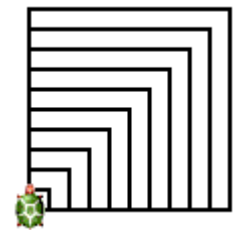
```
viem štvorec :DĺžkaStrany
opakuj 4 [ dopredu :DĺžkaStrany vpravo 90 ]
koniec
```

Táto procedúra začína klasicky kľúčovým slovom **viem** po ktorom nasleduje meno procedúry. Lenže po mene procedúry je tam novinka – procedúre sa dáva na známosť, že bude nasledovať obsah premennej, ktorú sme si nazvali **DĺžkaStrany**. (Mohli sme ju kľudne nazvať **ZlostnýŠakaVyjúciNaMesiac** alebo **x**, ale je dobre voliť názov tak, aby vypovedal o tom, čo do premennej ukladáme.) Obsah premennej **DĺžkaStrany** potom použijeme pri kreslení štvorca.

Keď naučíte Imagine uvedenú procedúru, môžete napísať **štvorec 100**. Číslo 100 sa uloží do premennej **DĺžkaStrany** a procedúra vám nakreslí pekný štvorček so stranou 100.

Úloha č.1: urobte si procedúru **štvorec** a s jej pomocou nakreslite to, čo vidíte na obrázku.

Parametre sa v procedúrach dajú využívať mnohými rôznymi spôsobmi. Môžeme ich použiť na určovanie dĺžok, uhlov, mien, tvarov, počtu opakovaní a iných vecí. O tom svedčia aj nasledujúce úlohy:



Úloha č.2: Urobte si nasledujúcu procedúru a spustite ju s parametrami 1 až 10. Čo to bude robiť? Prečo?

```
viem chaos :n
opakuj 11 [ dopredu 100 vpravo :n * 360 / 11 ]
koniec
```

⁶ Parameter je to číslo, ktoré za procedúrou nasleduje a ktoré určuje, ako sa procedúra bude chovať. Sú procedúry, ktoré môžu mať parametrov viacero – napríklad procedúra **urob** má dva. Meno premennej a jej novú hodnotu.

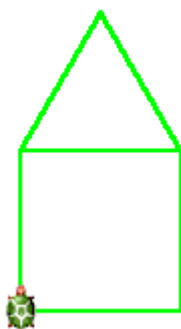
Úloha č.3: Urobte si nasledujúcu procedúru a spustite ju s nejakými vhodnými parametrami. Čo to bude robiť? Prečo?

```
viem divdivuci :n :d  
opakuj :n [ dopredu :d vpravo 360 / :n ]  
koniec
```

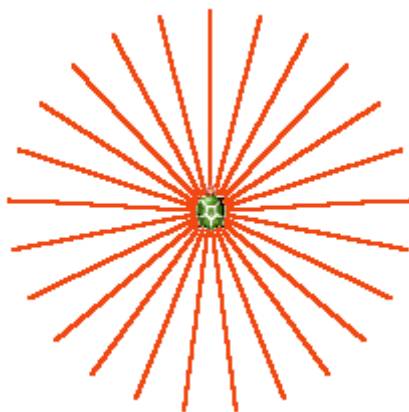
Úloha č. 4: Napíšte procedúru schody s jedným parametrom, ktorý bude udávať počet schodov, ktorý má korytnačka nakresliť. Ak bude mať parameter hodnotu 3, bude to vyzeráť takto:



Úloha č. 5: Napíšte procedúru domček, ktorá nakreslí domček určenej veľkosti. Takže napríklad domček 80 nakreslí toto:



Úloha č. 6: Napíšte procedúru hviezdička, ktorá nakreslí čiarkovú hviezdičku s toľkými čiarkami, koľko jej povieť s pomocou parametra. Napríklad hviezdička 25 urobí



(Možno sa vám bude hodiť príkaz vzad. Korytnačka má aj spiatočku, len jej treba povedať, o koľko má cúvnuť. Teda napríklad vzad 35.)

Úloha č. 7: Upravte príkaz hviezdička tak, aby mal dva parametre. Druhý parameter bude určovať, aké dlhé majú byť čiarky. (Predošlý obrázok by sa nakreslil príkazom hviezdička 25 100 .)

Piata kapitola

Rozhodná korytnačka

Doteraz sme od korytnačky Žofky vyžadovali bezpodmienečnú poslušnosť. Povedali sme jej, aké príkazy má vykonať a ona ich – ako sa na cvičené zviera patrí – poslušne vykonala. Tým sme ale hrubo podcenili jej potenciál. Korytnačka totiž nie je úplne slaboduchá a ak jej necháme podrobné pokyny, nemusí vždy vykonávať príkazy jeden za druhým, ale vie sa zariadiť podľa okolností. Dokonca vie nejaké veci nechať na náhodu. A v tejto kapitole si ukážeme, ako tieto jej vlastnosti využiť.

Začneme s tou náhodou. Korytnačka pozná príkaz `náhodne`. Ak jej niekde napíšeme `náhodne 100`, znamená to „vymysli číslo, ktoré je nezáporné a menšie ako sto“. Vyskúšajte jej niekoľkokrát za sebou zadať príkaz

`piš náhodne 100`

a pozerajte, aké čísla bude písať.

Náhodné čísla sa dajú využiť rôznymi spôsobmi, z ktorých si niekoľko ukážeme. Skúste napríklad zadať príkaz

`opakuj 5000 [vpravo náhodne 360 dopredu 10]`

Keď napíšete niečo ako `vpravo náhodne 360`, korytnačka sa otočí v pravo o nejaké náhodné číslo od 0 do 359 a získa tak úplne náhodný smer. A keď ju vždy pred ďalším krokom takto zmätiete, jej cesta bude pripomínať cestu opilca cez štadión (ako uvidíte, keď si uvedený príkaz vyskúšate).

Dobre. Teraz už nemáme všetko tak úplne pod kontrolou, korytnačka si nejaké čísla vymýšľa. Ešte ju treba naučiť, ako zareagovať na nejaké vonkajšie podmienky. A na to slúži príkaz `ak`. Používa sa napríklad takto:

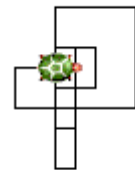
`ak :a = 1 [piš [V premennej a je jednotka.]]`

Za slovíčkom `ak` nasleduje podmienka a po nej hranatá zátvorka s príkazmi. Ak je podmienka splnená (teda v našom prípade, ak je v premennej `a` skutočne 1), príkazy v hranatej zátvorke sa vykonajú a objaví sa nápis. Ale ak podmienka splnená nie je, hranatá zátvorka sa preskočí a pokračuje sa až za ňou. (V našom prípade nie je kde pokračovať a príkaz nevypíše nič.)

Príkaz `ak` teraz využijeme na rafinované náhodné kreslenie. Príkaz `náhodne 2` nám vráti buď 0 alebo 1. A náš príkaz vymyslíme tak, aby vtedy, keď to bude jednička, korytnačka zahla vpravo, inak išla rovno. Bude to vyzerať takto:

`opakuj 500 [ak náhodne 2 = 1 [vpravo 90] dopredu 10]`

Obrázok, ktorý teraz korytnačka nakreslí, bude síce tiež náhodný, ale oveľa usporiadanejší (asi ako ulice v New Yorku).



Ak chceme, aby sa korytnačka točila nielen vpravo, ale aj vľavo, musíme príkaz trochu pozmeniť. Jednak musíme použiť `náhodne 3` miesto `náhodne 2`, pretože už máme tri možnosti: 0 – rovno, 1 – vpravo a 2 – vľavo. Okrem toho si výsledok funkcie `náhodne` budeme ukladať do premennej. A nakoniec, keďže je to väčší príkaz, spravíme si z neho samostatnú procedúru.

viem NewYork

`opakuj 500 [`

`urob "náhoda náhodne 3`

`ak :náhoda = 1 [vpravo 90]`

```

    ak :náhoda = 2 [ vľavo 90 ]
    dopredu 10
  ]
koniec

```

Úloha č.1: Vyskúšajte si všetky uvedené príklady. Čo to spraví, keď v procedúre NewYork nahradíte príkazy vpravo 90 a vľavo 90 príkazmi vpravo 120 a vľavo 120 ?

Úloha č.2: Napíšte procedúru kocka, ktorá vám vypíše náhodné číslo od 1 do 6.

Úloha č.3: Nasledujúca procedúra zistí, či číslo, ktoré dostala ako parameter je párne, a ak je, vypíše to.

```

viem parita :n
  ak zvyšok :n 2 = 0
    [ (píš [Číslo] :n [ je párne. ] ) ]
koniec

```

Funkcia zvyšok x y vypočíta zvyšok po delení čísla x číslom y , takže napríklad zvyšok 14 3 bude 2 (pretože $14 : 3 = 4$ zvyšok 2.) Upravte procedúru parita tak, aby vedela rozoznať párne čísla od nepárnych a aby pri každom čísle vedela napísať, aké vlastne je.

Úloha č.4: Čo bude robiť nasledujúca procedúra?

```

viem čudo :počet
  urob "iks 1
  opakuj :počet [
    ak zvyšok :iks 2 = 0 [ vpravo 90 ]
    ak zvyšok :iks 2 = 1 [ vľavo 90 ]
    dopredu 20
    urob "iks :iks + 1
  ]
koniec

```

Skúste spustiť čudo 7 alebo čudo 12. Ako to funguje? Urobte podobnú procedúru, ktorá ak dostane na vstupe 9, nakreslí toto:



Úloha č.5: Čo bude robiť nasledujúca procedúra?

```

viem záhada :a
  ak :a <= 0 [ ukonči ]
  dopredu :a
  vpravo 90
  záhada :a - 1
koniec

```

Spustíte ju napríklad záhada 20 alebo záhada 200. Ako to vlastne funguje?

Šiesta kapitola

Korytnačka s perspektívou

Korytnačka Žofka sa stáva čoraz zodpovednejšou. Vie si zistiť stav vecí okolo seba a s pomocou príkazu `ak` vie naň zareagovať. Príkaz `ak` je ale relatívne jednoduchý. Vieme s jeho pomocou korytnačke vysvetliť, že `ak` sa niečo udeje, má sa správať určitým spôsobom. Ale nevieme do príkazu zahrnúť, ako sa má správať, ak sa to niečo neudeje. Korytnačka môže mať pred sebou veľké množstvo možností a z nich si môže chcieť vybrať len jednu. V tejto kapitole si ukážeme niektoré vylepšenia príkazu `ak` s pomocou ktorých bude dorozumievanie s korytnačkou jednoduchšie.

Prvým z týchto vylepšení je príkaz `ak2`. Ak ste použili normálne `ak`, musela po ňom nasledovať podmienka a potom v hranatých zátvorkách príkazy, ktoré sa majú vykonať, ak je podmienka splnená. Ak použijete príkaz `ak2`, po podmienke musia nasledovať dve sady príkazov. Každá musí byť uzavretá v hranatých zátvorkách. Prvá z nich sa vykoná, ak je podmienka splnená, druhá sa vykoná, ak podmienka splnená nie je. Ako to funguje, si ukážeme na procedúre `mnk` (mimoriadne nadaná korytnačka).

```
viem mnk
  píš [Napíš číslo:]
  urob "číselko čítajSlovo
  ak2 zvyšok :číselko 2 = 0
    [ (píš [Číselko ] :číselko [je párne.]) ]
    [ (píš [Číslíčko ] :číselko [je nepárne.]) ]
koniec
```

Táto procedúrka používa niektoré nové príkazy. Príkaz `čítajSlovo` počká, kým niekto nenapíše niečo do riadku a nestlačí `<Enter>`. To si uložíme do premennej `číselko`. Funkcia `zvyšok` vie vypočítať zvyšok po delení. Keď napríklad napíšete `píš zvyšok 4275 7`, dozviete sa zvyšok po delení čísla 4275 siedmimi. Túto funkciu použijeme na to, aby sme zistili, či je číslo uložené v premennej `číselko` párne. (Párne čísla dávajú pri delení dvoma zvyšok 0.) Ak číslo párne je, korytnačka oznámi, že párne bolo. Inak dá vedieť, že párne nie je. Skrátka – mimoriadne nadaná korytnačka.

Úloha č.1: Vyskúšajte si to.

Druhé vylepšenie prichádza ku slovu, ak máme možností viacero. Predstavte si, že primitívny africký kmeň si u vás objednal software na prevod z všeobecne používanej číselnej sústavy do tej ich. Tento kmeň pozná ale iba tri čísla – jeden, dva a tri. Akékoľvek iné číslo je pre nich „veľa“. Siahneme po `Imagine`, dohodneme sa s autormi na províziách a vytvoríme nasledujúcu procedúru:

```
viem zulukafer :čís
  akJe :čís
  [
    1 [píš "Jeden]
    2 [píš "Dva]
    3 [píš "Tri]
    [píš "Veľa]
  ]
koniec
```

Po príkaze **akJe** tentokrát nenasleduje podmienka, ale hodnota. A v hranatých zátvorkách sú potom vymenované všetky prípady, ktoré pripadajú do úvahy. Každý prípad má pri sebe pribalené svoje vlastné malé hranaté zátvorčičky, v ktorých je kód, ktorý sa má vykonať, keď daný prípad nastane. Na koniec môžeme uviesť hranatú zátvorku s kódom, ktorý sa vykoná, ak žiadna z možností nenastala. Tak, a môžeme v danom kmeni otvoriť pobočku našej softwarovej firmy.

Úloha č.2: Vyskúšajte si to. (To programovanie, nie otváranie pobočky.)

Elegantnejšie využitie príkazu **akJe** môžete vidieť na nasledujúcej procedúre:

```
viem behaj
  akJe kláves
  [
    vpravo [ vpravo 5]
    vľavo  [ vľavo 5]
    hore   [ dopredu 5]
    Esc    [ ukonči ]
  ]
  behaj
koniec
```



Funkcia **kláves** počká na stlačenie klávesu a povie, ktorý to bol. Ak to bola šípka vpravo, korytnačka sa pootočí doprava o 5 stupňov. Ak to bola šípka vľavo, otočí sa doľava. Ak šípka hore, pohne sa a ak stlačíte klávesu <Esc>, procedúra sa skončí. Inak sa funkcia **kláves** zavolá znova a tak sa stále opakuje. Po spustení tejto funkcie môžete korytnačku pilotovať z klávesnice.

Úloha č.3: Dorobte do tejto procedúry ďalšie ovládacie klávesy na zmenu hrúbky alebo farby čiary, na zdvihnutie a položenie pera, na cúvanie...

Aby korytnačka nemala tej slobody zas príliš, postavíme jej ohradu (teda postaví si ju sama). Urobte jej na ňu procedúrku (ohrada by mala byť červená a vyzerá ako na obrázku) a presťahujte korytnačku do nej. Nezabudnite potom zmeniť hrúbku a farbu pera na pôvodné.



Predstavte si, že chceme používať náš kresliaci program, ale pri tom nechceme, aby nám korytnačka ušla z ohrady. Musíme preto príkaz **dopredu 5** z procedúry **behaj** nahradiť niečím „opatnejším“, niečím čo dá pozor, aby sme si ohradu nepočmárali, ani z nej nevybehli. Najprv sa pôjdeme pozrieť, či na tom mieste, kam sa chystáme, nie je už náhodou červená – teda ohrada. (Samozrejme predtým zrušíme kreslenie.) Ak zistíme, že tam, kam sa chystáme, je bezpečno, vrátime sa naspäť a potiahneme tam čiaru. Ak tam bezpečno nie je, iba sa vrátime naspäť. Na zistenie farby bodu na ktorom korytnačka stojí, slúži príkaz **farbaBodu**. Takže varianta „hore“ z príkazu **behaj** bude teraz vyzerá takto:

```
hore [ ph dopredu 5
      ak2 farbabodu = "červená
        [vz 5 pd]
        [vz 5 pd do 5]
      ]
```

Úloha č.4: Vyskúšajte si to.

Úloha č.5: Upravte program náhodného pohybu korytnačky z minulej lekcie tak, aby korytnačka neušla z ohrady

Úloha č.6: Čo spraví príkaz

```
opakuj 4000 [ ak2 náhodne 2 = 0 [vp 90] [vl 90] do 10 ]
```

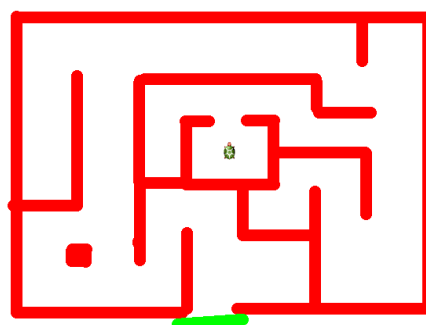
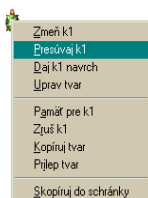
Úloha č.7: Napíšte procedúru, ktorá si vypýta s pomocou **čítajSlovo** z klávesnice číslo a slovne vypíše jeho zvyšok po delení tromi (teda napíše nula, jeden alebo dva). Použite príkaz **akJe**.

Siedma kapitola

Vytrvalá korytnačka

Isto je vám známy príbeh o Minotaurovi, ktorý sídlil v Labyrinte na Kréte. Jeho likvidácia v sebe zahŕňala dva problémy. Prvý bol samotné zlikvidovanie príšery, ktorá bola napolo človek, napolo hovädzí dobytok. Druhý problém nastal hneď vzápätí. A to dostať sa z Labyrintu zase von. Grécke báje tvrdia niečo o princeznej Ariadne a o kľbku. Táto verzia má ale svoje slabiny. Stačí si len predstaviť klasického gréckeho bojovníka akým určite bol Théseus – v ľavej ruke štít, v pravej meč, v zuboch kľbko, ako vzdoruje strašnému netvorovi. Poriadna akčná scéna, akou bolo zabíjanie obludy v sebe zahŕňa skoky z výšok, prenasledovanie a ukrývanie sa v tmavých chodbách a výklenkoch. Je ťažké uveriť, že popri tom všetkom má kladný hrdina ešte čas rozmotávať špagát. Oveľa pravdepodobnejšie je, že Théseus mal skrátka šťastie a dosť vytrvalosti na to, aby našiel cestu von.

Predstavme si na chvíľu, že sa do podobnej situácie dostala korytnačka Žofka. S Minotaurom si hravo poradila a teraz hľadá cestu. Aby sme túto situáciu znázornili čo najdokonalejšie, načítame si nejaké vhodné pozadie. To pozadie najprv musíme mať. Použijeme obrázok bludisko.bmp, ktorý uložíme do patričného adresára (podadresár Obrázky adresára s Imaginom). Obrázok do pozadia stránky č. 1 (to je tá stránka, s ktorou sme doteraz vždy pracovali) vložíme príkazom



Stránka1'nechPozadie "bludisko.bmp

Korytnačka nebude celkom presne v strede labyrintu, tak na ňu klikneme pravým tlačidlom myši, vyberieme možnosť *Presúvaj* a potiahneme ju tam. Situácia by mala byť podobná, ako na prvom obrázku.

Steny labyrintu tvorí červená farba. Na blúdenie teda môžeme použiť fintu z predošlej kapitoly – urobíme opatrný krok bez kreslenia náhodným smerom, ak je tam červeno, ujdeme odtiaľ, inak sa tam presunieme. Toto budeme opakovať stále dokolečka. Problém ale je, dokedy to máme opakovať. Doteraz používaný príkaz *opakuj* sa do tejto situácie rozhodne nehodí. Nikde nemáme zaručené, že korytnačka dosiahne zelený prah bludiska po presne tisíc alebo presne milión krokoch. Nevieme, koľkokrát to treba zopakovať, ale chceme to opakovať dovtedy, kým sa niečo nestane. A na to presne je určený nový typ cyklu riadeného podmienkou – príkaz *kým*.

Za príkazom *kým* nasledujú dva úseky uzavreté v hranatých zátvorkách. Prvý je podmienka.⁷ Druhý sú príkazy, ktoré sa budú opakovať, *kým* je podmienka splnená. Takže procedúra na blúdenie bludiska by mohla vyzeráť takto:

```
viem Blúd'
  urob "smeDoma "nie
  kým [ :smeDoma = "nie ]
  [
    vpravo náhodne 360
    peroHore
    dopredu 10
```

⁷ Príkaz *kým* má okolo podmienky hranaté zátvorky. Príkaz *ak* má hranaté zátvorky okolo podmienky nesmie!!!

```

    ak farbaBodu = "zelená
      [ urob "smeDoma "áno ]
    ak2 farbaBodu = "červená
      [ vzad 10 ]
      [ vzad 10 peroDolu dopredu 10 ]
  ]
koniec

```

Celý cyklus sa bude opakovať dovtedy, kým premenná `smeDoma` bude mať hodnotu `nie`. Korytnačka sa bude náhodne potulovať po labyrinte dovtedy, kým nenarazí na zelenú farbu. Vtedy sa hodnota premennej `smeDoma` zmení na `áno`, podmienka prestane byť splnená a cyklus sa skončí.

Úloha č.1: Vyskúšajte to.

V tomto prípade má Žofka oproti Théseovi jednu nevýhodu a jednu výhodu. Jej nevýhodou je, že je pomerne hlúpučká (alebo nie dosť šikovne naprogramovaná). Potáca sa po labyrinte ako slepá a nijako sa nesnaží ísť správnym smerom. Všetko nechá na náhodu. Jej výhodou je, že je veľmi rýchla. Takže aj s takýmto mizerným algoritmom raz k cieľu dôjde.

Úloha č.2: Čo bude robiť nasledujúca procedúra?

```

viem otravka
  piš [ Zadaj heslo: ]
  urob "vstup čítajSlovo
  kým [ :vstup <> "orangutan ]
  [
    piš [ Nesprávne heslo. Zadaj heslo: ]
    urob "vstup čítajSlovo
  ]
  piš [ Vitaj. ]
koniec

```

Úloha č.3: Čo bude robiť nasledujúca procedúra? Skúste na to prísť napriek tomu, že obsahuje viacero nových funkcií korytnačky. Príkaz `bodka` spraví na danom mieste nastavenou farbou a perom bodku. Podmienka `kláves?` je splnená, ak bol stlačený niektorý kláves, ktorý ešte nebol spracovaný. Kontrolujeme to, aby nám program nezastal na príkaze `kláves` a nečakal, kým sa niečo stlačí, ale aby bežal ďalej. (Môžete tú kontrolu skúsiť vynechať a pozrite sa, čo to bude robiť potom.) Príkaz `čakaj 1` spôsobí, že program zastane na jednu tisícinu sekundy.

```

viem beh
  nechFp "zelená nechHp 10 bodka
  peroHore vpravo náhodne 360
  kým [ farbaBodu = "zelená ] [ dopredu 1 ]

```

```

kým [ farbaBodu <> "zelená ]
[
  ak kláves?
  [
    akJe kláves
    [
      vpravo [ vpravo 3 ]
      vľavo [ vľavo 3 ]
      esc [ ukonči ]
    ]
  ]
  dopredu 1 čakaj 1
]
koniec

```

Úloha č.4: Napíšte procedúru, ktorá nebude robiť nič až kým nestlačíte šipku hore. Vtedy odštartuje korytnačku a posunie ju o 1000 krokov dopredu.

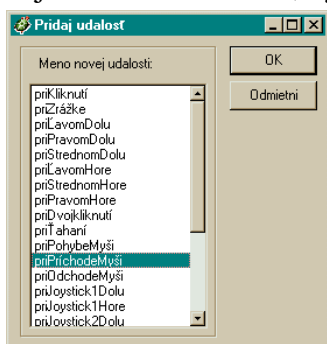
Úloha č.5: Kdesi náhodne na obrazovke urobte zelenú bodku. Potom korytnačku pošlite domov a natočte o jeden stupeň vpravo. Vymyslite procedúru, ktorá nechá korytnačku ísť dopredu až dotedy, kým tú zelenú bodku nenájde. Tam korytnačka zastane. (Dá sa to spraviť aj s pomocou kým, aj s pomocou procedúry, ktorá zavolá sama seba. Vyskúšajte obe možnosti.)

Ôsma kapitola

Vnímavá korytnačka

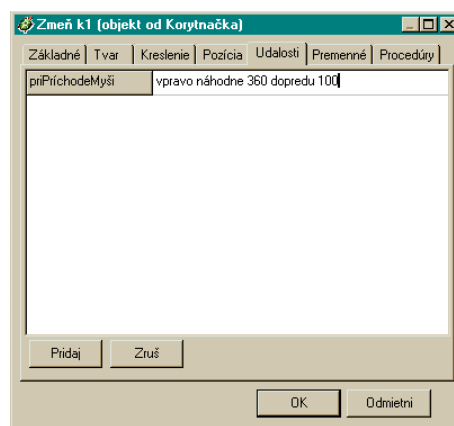
Korytnačka Žofka sa vie čoraz lepšie prispôbiť ťažkým požiadavkam, ktoré sa na ňu kladú. Vie reagovať na okolité podmienky, vie sa rozhodovať, je vytrvalá a jej inteligencia a rýchlosť jej dávajú veľké predpoklady prežiť v dnešných ťažkých časoch (aj za tú cenu, že bludisko, ktoré by každý prváčik vyblúdil za dvadsať sekúnd, blúdi Žofka hodinu).

Korytnačkine možnosti však zďaleka nie sú vyčerpané. Korytnačka okrem toho, že vykonáva nejaký príkaz, môže dávať pozor, či sa jej neprihodila nejaká zvláštna udalosť. Vie si zistiť, že ponad ňu prešla myš, že na ňu klikli niektorým tlačidlom myši, že niekto stláča streľbu na joysticku alebo že sa zrazila s inou korytnačkou.⁸ A my jej môžeme povedať, aby v prípade, že nejaká udalosť nastala, vykonala určitú činnosť.



Otvorte si okno *pamäť*, rozbaľte stránku *Stránka1*, nájdite v nej korytnačku (pravdepodobne má prozaické meno *k1*) a dvojkliknite na ňu. Rozbaľí sa vám dialógové okno, v ktorom sa dajú nájsť všetky údaje o korytnačke. Vy si vyberiete kolónku *udalosti*. V tejto kolónke toho zatiaľ veľa nenájdete – stlačte teda tlačidlo *Pridaj*. Imagine na vás vychrlí všetky možné udalosti, na ktoré vie korytnačka zareagovať. Vyberte si udalosť *priPríchodeMyši*

a stlačte *OK*. Udalosť sa nám objaví v okne udalostí. Znamená to, že korytnačka odteraz dáva pozor, či nad ňu myš nepríde. Akonáhle sa tak stane, korytnačka vykoná príkazy, ktoré jej napíšeme vedľa udalosti. My jej tam dáme



vpravo náhodne 360 dopredu 100

Od tejto chvíle korytnačka začne byť alergická na myš. Akonáhle nad ňu prídeme s kurzorom myši, naberie náhodný smer a uskočí o 100 krokov.

Úloha č.1: Vyskúšajte si to.

Ak chceme korytnačku z tejto alergie vyliečiť, treba sa v okne *Udalosti* nastaviť na patričnú udalosť a stlačiť tlačidlo *Zruš*.

Celý opísaný proces nemusíme robiť len tak, že otvárame okienka a niečo tam vypisujeme. Obsluhu udalostí môžeme nastavovať aj z príkazového riadku a – čo je dôležitejšie – aj priamo z niektorej procedúry vášho programu. Ak by sme chceli zmeniť paniku korytnačky tým spôsobom, že pred myškou bude utekať dookola do štvorca, môžeme zadať príkaz

k1'nechUdalosť "priPríchodeMyši [vpravo 90 dopredu 100]

A týmto sme sa dostali k ďalšiemu dôležitému znaku Imaginu. Po úvodzovkách (*"*), ktorými sa začínajú reťazce a dvojbodke (*:*), ktorú treba dať pred meno premennej, aby sme sa dostali k tomu, čo je v nej uložené je to apostrof (*'*). Používa sa vtedy, ak potrebujeme povedať, že niečo sa netýka hlavného okna, ale niektorého iného objektu. Keby sme napísali iba

nechUdalosť "priPríchodeMyši [vpravo 90 dopredu 100]

Imagine by si myslel, že má akciu [vpravo 90 dopredu 100] vykonať pri príchode myši nad

⁸ Nebojte sa, korytnačka vám do susedného počítača neujde, aby sa tam zrazila s tamojšou. Ale korytnačka nemusí byť na ploche jediná – môže ich byť viacero. O tom ale niekedy nabudúce.

hlavné okno a bol by zmätený, lebo napríklad príkaz `dopredu 100` je určený pre korytnačky a nie pre okná. Treba mu preto povedať, že príkaz `nechUdalosť`, ktorým sa nastavuje, čo sa má pri danej udalosti vykonať sa netýka hlavného okna, ale korytnačky k1. A to sa spraví práve tak, že sa zavolá procedúra `k1'nechUdalosť`. Nezabudnite na úvodzovky pred menom udalosti.

Keď sme už raz otvorili okno s údajmi o korytnačke, poďme sa pozrieť aj na ďalšiu kolónku – *premenné*. Doteraz sme pracovali iba s premennými, ktoré boli globálne. Každá procedúra s nimi mohla narábať a boli viditeľné od všadiaľ. Ale korytnačka môže mať svoje vlastné premenné, ktoré o nej budú voľačo hovoriť. Ak chceme korytnačke k1 zriadiť premennú `Rýchlosť`, spravíme to príkazom `nechVlastná`:

```
k1'nechVlastná "Rýchlosť 5
```

V okne *pamäť* sa pri korytnačke objaví chlievik s premennými a našu novú premennú si tam môžeme nájsť.

Teraz korytnačke povieme, aby si vždy, keď na ňu klikneme, zmenšila rýchlosť o 1 a zmenila náhodne farbu pera.

```
k1'nechUdalosť "priKliknutí [ nechRýchlosť Rýchlosť – 1 nechFp náhodne 16 ]
```

Isto ste si všimli, že súkromné premenné sú istým spôsobom privilegované – nepíšu sa pri nich dvojbodky a keď chceme zmeniť hodnotu v premennej `Rýchlosť`, máme na to dokonca nový špeciálny príkaz `nechRýchlosť`. Na druhú stranu musíme dávať pozor, aby sme premennú používali buď vo vlastných procedúrach korytnačky alebo sa na ne odvolávali s pomocou apostrofu. Môžeme teda vytvoriť takúto procedúru

```
viem chyťMa
  k1'nechRýchlosť 5
  kým [ k1'Rýchlosť > 0 ]
  [
    dopredu k1'Rýchlosť
    vpravo náhodne 360
    čakaj 5
  ]
koniec
```

Keby korytnačka neznižovala svoju rýchlosť pri kliknutí, cyklus z tejto procedúry by nikdy neskončil a korytnačka by pobežovala po ploche kým by nevypadli poistky. Našťastie korytnačka na klikanie pozor dáva. Treba na ňu kliknúť myšou päťkrát, aby sa zastavila úplne. Prajeme príjemnú zábavu.⁹

Úloha č.2: Vyskúšajte to.

Úloha č.3: Vylepšite korytnačku tak, že keď na ňu dvojkliknete, spraví okolo seba zelený štvorec.

Úloha č.4: Čo s korytnačkou k1 spraví nasledujúci príkaz?

```
k1'nechUdalosť "priKliknutí [ ak2 pero = "pd [peroHore] [peroDole] ]
```

Úloha č.5: Zriadte korytnačke súkromnú premennú `Odpočet` v ktorej bude na začiatku číslo 10. Pri každom prejdení myši nad korytnačkou sa hodnota v nej buď zmenší o jedna, alebo ak je už nulová, korytnačka vyrazí náhodným smerom o 300 korytnačích krokov a `Odpočet` sa znovu nastaví na 10.

Úloha č.6: Vyrobte korytnačke súkromnú premennú `Strana`. Keď na korytnačku kliknete ľavým tlačidlom, nakreslí štvorec so stranou dĺžky `Strana`. Keď kliknete pravým tlačidlom, `Strana` sa zväčší o 10 a keď stredným (alebo kolieskom), `Strana` sa o 10 zmenší.

⁹ Ak ste náhodou zabudli zrušiť to, že korytnačka pred myšou uteká, spravte to teraz. Inak sa vám nepodarí na korytnačku kliknúť.

Deviata kapitola

Vypočítavá korytnačka

V tejto kapitole sa ešte nakrátko vrátíme k procedúram. Všetky procedúry, ktoré sme doteraz napísali, boli určené na to, aby niečo robili. Ale existujú aj iné procedúry. Napríklad procedúra `zvyšok`, ktorú sme tak často a s úspechom použili, nemá za úlohu niečo robiť, ale niečo vypočítať. Dôležitý nie je vonkajší efekt, ale výsledok, ktorý z procedúry vypadne. Ako sa takéto procedúry robia, si ukážeme na nasledujúcom príklade, ktorý je síce nezmyselný, ale zato poučný.

```
viem súčet :a :b
výsledok :a + :b
koniec
```

Príkaz `výsledok`¹⁰ zoberie to, čo mu predhodíte ako parameter a vyhlási to za výsledok celej procedúry. Takže ak by ste zadali príkaz

```
píš súčet 1 1
```

Imagine vám napíše 2. (Pozor! Príkaz `píš` je tam dôležitý. Keby ste napísali iba `súčet 1 1`, Imagine by síce vypočítal, že 2, ale nevedel by, čo s tou dvojku má robiť a sťažoval by sa.) Akonáhle sa procedúra dopracuje k príkazu `výsledok`, tak skončí. Takže ak by ste vyššie uvedenú procedúru upravili nasledujúcim spôsobom:

```
viem súčet :a :b
výsledok :a + :b
píš "Kikirikí
koniec
```

k žiadnemu kikiríkaniu nedôjde. Príkaz `výsledok` má rovnaký efekt, ako príkaz `ukonči`.

Predstavte si, že chceme napísať procedúru, ktorá nám nájde najmešieho rozumného deliteľa daného čísla. (Najmenší deliteľ je vždy číslo 1, takže „rozumný“ deliteľ bude taký ktorý je aspoň 2.) Začneme tak, že si do premennej `b` vložíme dvojku. Pozrieme sa, či je kontrolované číslo deliteľné `b`-čkom.¹¹ Ak je, povieme, že je to výsledok. Ak nie je, zvýšime `b`-čko o 1 a zase skontrolujeme. Tak budeme `b`-čko stále zvyšovať a skúšať až dovtedy, kým nejaké vhodné číslo nenájdem. Takže procedúra by mohla vyzeráť napríklad takto:

```
viem deliteľ :a
lokálne "b
urob "b 2
kým [ :b <= :a ]
[
    ak zvyšok :a :b = 0
    [ výsledok :b ]
    urob "b :b + 1
]
koniec
```

V tejto procedúre sú povšimnutiahodné dva detaily. Prvý je podmienka cyklu `kým`. Cyklus sa bude

¹⁰ Skratka pre príkaz `výsledok` je `vy`.

¹¹ Najlepšie to zistíme tak, že vypočítame zvyšok kontrolovaného čísla pri delení `b`-čkom a ak je ten zvyšok nula, číslo deliteľné je.

opakovať (a b-čko zväčšovať) iba dovtedy, kým je b-čko menšie alebo rovné ako kontrolované číslo. Vzhľadom na to, že kontrolované číslo bude určite deliteľné aspoň samo sebou, stačí skúšať iba po neho. Druhý zaujímavý detail je príkaz lokálne "b. Isto ste si všimli, že keď v procedúre používate nejakú premennú, objaví sa vám v okne *pamäť* medzi globálnymi premennými. To je niekedy užitočné. Ale niekedy (ako napríklad v tejto procedúre) potrebujeme použiť nejakú premennú len na pomocný výpočet a medzi globálnymi premennými by nám zavádzala. Tak na začiatku povieme, že tá premenná je *lokálna*, teda, že je vytvorená iba pre danú procedúru. Nikto iný okrem danej procedúry túto premennú nevidí a nemôže spraviť žiadnu škodu tým, že by omylom zmenil jej hodnotu.

Úloha č.1: Napíšte procedúru *deliteľ* a vyskúšajte ju (napríklad píš *deliteľ 221*)

Ďalšia vec, ktorú si na vypočítavých procedúrach ukážeme, je finta, ktorá sa volá *rekurzia*. Túto fintu sme už niekoľkokrát použili, ale neupozorňovali sme na to. V budúcnosti o nej bude celá kapitola. Predstavte si, že chcete napísať procedúru *faktoriál*, ktorá vypočíta – čuduj sa svete – faktoriál daného čísla. (Pre nezasvätených – ak chcem vypočítať napríklad „7 faktoriál“, musím vypočítať $1.2.3.4.5.6.7 = 5040$, „5 faktoriál“ je $1.2.3.4.5 = 120$) Využijeme na to tú vlastnosť, že ak poznám napríklad „7 faktoriál“, tak vypočítať „8 faktoriál“ je už relatívne jednoduché. Stačí „7 faktoriál“ vynásobiť ôsmimi. Takže procedúru napíšeme nasledujúcim ulieváckym spôsobom:

```
viem faktoriál :n
  ak2 :n = 1
    [ výsledok 1 ]
  [ výsledok :n * faktoriál (:n - 1) ]
koniec
```

Čo to spraví, ak zadáme príkaz *píš faktoriál 3* ?

- Zavolá sa funkcia *faktoriál 3*. Skontroluje sa, či je :n jednotka (nie je, lebo je tam trojka) a tak výsledok funkcie bude $3 \times$ faktoriál 2. Problém je, že Imagine zatiaľ netuší, čomu sa rovná faktoriál 2. A tak funkciu *faktoriál* zavolá znovu.
 - Zavolá sa funkcia *faktoriál 2*. Skontroluje sa, či je :n jednotka (stále nie je, je tam dvojka) a tak výsledok funkcie bude $2 \times$ faktoriál 1. faktoriál 1 však stále nevieme, čo je. A tak funkciu *faktoriál* voláme ešte raz.
 - Zavolá sa funkcia *faktoriál 1*. Skontroluje sa, či je :n jednotka. Je. Takže výsledok bude 1.

Teraz už Imagine vie, že faktoriál 1 je 1. Môže teda vypočítať $2 \times$ faktoriál 1 (bude to 2) a vrátiť to ako výsledok faktoriál 2.

Teraz už Imagine pozná aj faktoriál 2. Môže teda vypočítať $3 \times$ faktoriál 2 a vrátiť 6 ako výsledok faktoriál 3

Funkcia *píš* tú šestku vypíše.

Keby sme zadali príkaz *píš faktoriál 8*, tak sa funkcia *faktoriál* zavolá sama seba podobným spôsobom osemkrát. Takáto procedúra – to už je niečo, čo už je dosť elegantné na to, aby ste sa tým mohli pochváliť pred príbuzenstvom. Väčšina síce nebude rozumieť, ale znalci ocenia.

Úloha č.2: Vyskúšajte to. (Nie to chválenie sa. Tú procedúru.)

Úloha č.3: Napíšte procedúru *menšie :a :b*, ktorá dostane na vstupe dve čísla a ako výsledok vráti to menšie z nich.

Úloha č.4: Napíšte procedúru `kocka2`, ktorá ako svoj výsledok bude dávať náhodné číslo od 1 do 6 (Áno, dobre sa pamätáte, taká podobná procedúra už bola.) Sľ ažená varianta pre machrov – napíšte procedúru `náhoda` `:u` `:v` s dvoma vstupnými parametrami, ktorá vráti náhodné číslo od `:u` do `:v`.

Úloha č.5: Napíšte procedúru `priemer` `:a` `:b`, ktorá vypočíta priemer dvoch čísel.

Úloha č.6: Napíšte procedúru `súčet2` `:a`, ktorá spočíta všetky čísla od 1 do `:a`. Takže napríklad `súčet2 5` bude $1 + 2 + 3 + 4 + 5 = 15$.

Úloha č.7: Napíšte procedúru `súčet3` `:a`, ktorá spočíta všetky nepárne čísla menšie alebo rovné `:a`. Takže napríklad `súčet3 9` bude $1 + 3 + 5 + 7 + 9 = 25$.

Úloha č.8: (ľ ažká preľ ažká) S pomocou procedúry `deliteľ` napíšte procedúru, ktorá rozloží zadané číslo na prvočísla.

Desiata kapitola

Korytnačka a myš

Kapitola, do ktorej ste sa práve začítali, pojednáva o vzťahu dvoch zvierat – korytnačky Žofky a myši.¹² Oba tieto zvery veľmi úzko súvisia s počítačmi. Žofka je zviera virtuálne. Existuje len v počítači a celé dni len vysedáva na harddisku a čaká, kedy už konečne niekto spustí Imagine. Myš je naproti tomu zviera do istej miery reálne. Môžete ju chytiť do ruky, ak to práve nie je nejaký bezdrôtový mutant aj chvost má a tlačidlá na nej môžete stláčať aj vtedy, keď v okolí 100 kilometrov nie je žiaden počítač. Problém ale je, že bez počítača stláčanie tlačidiel na myši nemá žiadny zmysel. Pokiaľ myš nemôže pohybovať malou šipkou po termináli, jej život je pustý a prázdny.

Korytnačka aj myš môžu nažívať v celkom príjemnej symbióze. Korytnačka vie zistiť, kam myš práve ukazuje a zachovať sa podľa toho a myš je zase spokojná, že s jej pomocou môže užívateľ niečo korytnačke prikázať.

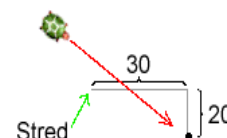
Ak si korytnačka chce zistiť, kde sa práve myš nachádza, slúži jej na to príkaz `pozMyši`. Skúste zadať príkaz

`kým [nieje kláves?] [piš pozMyši]`

Tento príkaz bude stále vypisovať, kde sa práve myš nachádza, až kým mu to nezatrhnete tak, že stlačíte nejaký kláves na klávesnici. Ako vidíte, „pozícia myši“ sú dve čísla. Prvé z nich hovorí, ako ďaleko je myš od stredu obrazovky vo vodorovnom smere, druhé hovorí, ako ďaleko je v zvislom smere. Ak je myš od stredu vpravo, je prvé číslo kladné. Ak je od stredu vľavo, je prvé číslo záporné. Podobne je to aj s druhým číslom. Ak je myš od stredu vyššie, je kladné a ak je myš od stredu nižšie, je záporné. Spustite si uvedený príkaz ešte raz a skúste nastaviť myš presne do stredu. (Malo by vám to vypisovať `0 0`. Buďte trpezliví, nie je to až také jednoduché.)

Dobre. Kde je myš už si zistiť vieme. Teraz to ešte nejakou využijť. Skúsime napísať procedúru, pri ktorej bude korytnačka pomaly liezť za myšou. Na to ale budeme potrebovať ešte dve nové veci. Prvou je príkaz `nechSmer`. Korytnačku sme doteraz otáčali iba príkazmi `vpravo` a `vľavo`. Nový smer vždy záležal od toho, kam bola korytnačka otočená predtým. Príkaz `nechSmer` naproti tomu nastaví korytnačku „podľa kompasu“. Po príkaze `nechSmer 0` bude korytnačka otočená na sever (hore), po príkaze `nechSmer 90` sa otočí na východ (vpravo), príkaz `nechSmer 180` ju otočí na juh (dolu) a príkaz `nechSmer 270` ju otočí na západ (doľava). Samozrejme môžeme použiť ľubovoľné iné číslo – odmeria sa patričný počet stupňov od smeru na sever v smere hodinových ručičiek a tým smerom sa korytnačka otočí. Ak by sme teda potrebovali otočiť korytnačku na juhovýchod, použijeme príkaz `nechSmer 135`.

Druhá novinka je príkaz `smerK`. Táto funkcia vie pre korytnačku vypočítať smer k nejakému bodu na ploche. Ak by sme si napríklad potrebovali zistiť smer k stredu plochy, vypočítali by sme to ako `smerK [0 0]`. Ak by sme potrebovali zistiť smer k bodu, ktorý je 30 korytnačích krokov vpravo od stredu a 20 krokov dole, smer bude `smerK [30 -20]`. (Pozrite si obrázok. Červená šípka ukazuje smer, ktorým bude korytnačka otočená, ak zadáme príkaz `nechSmer smerK [30 -20]`.) Ak chceme, aby sa korytnačka otočila smerom k bodu, na ktorý ukazuje myš, použijeme príkaz `nechSmer smerK pozMyši`.



Takže teraz to celé skúsme dať dohromady.

¹² Myš sa môže volať rôzne, napríklad Logitech alebo Microsoft.

```

viem prenasleduj
  kým [ nieje kláves? ]
  [
    nechSmer smerK pozMyši
    dopredu 1 čakaj 50
  ]
koniec

```

Úloha č.1: Vyskúšajte!

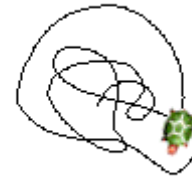
Procedúrka `prenasleduj` už robí takmer to, čo potrebujeme. Má ale jednu drobnú slabinu. Keď necháme myš stáť a korytnačka ju dobehne, začne nad ňou besne poskakovať. Je to spôsobené malou zákernosťou `Imaginu` – keď sa ho opýtame na pozíciu myši, napíše nám dve celé čísla. Ale skúste pohnúť korytnačkou (napríklad `vp 30 do 50`) a spýtať sa `Imaginu` na jej pozíciu (`píš poz`). Napíše vám dve desatinné čísla. Miesto, kde sa korytnačka nachádza, si totiž `Imagine` pamätá oveľa presnejšie. Lenže s takouto presnosťou je ťažké trafiť sa presne do miesta kde je myš a tak korytnačka poskakuje z jednej strany na druhú.

Úprava procedúry bude jednoduchá – pozrieme sa, či pozícia korytnačky je dosť ďaleko od pozície myši a niečo budeme robiť iba vtedy, ak je tá vzdialenosť aspoň jeden krok. Na to, aby sme zistili vzdialenosť dvoch bodov (konkrétne pozície korytnačky `poz` a pozície myši `pozMyši`) použijeme matematickú fintu zvanú `abs`. Takže upravená verzia bude vyzeráť takto:

```

viem prenasleduj
  kým [ nieje kláves? ]
  [
    ak abs (poz - pozMyši) > 1
    [
      nechSmer smerK pozMyši
      dopredu 1 čakaj 50
    ]
  ]
koniec

```



Úloha č.2: Vyskúšajte upravenú verziu.

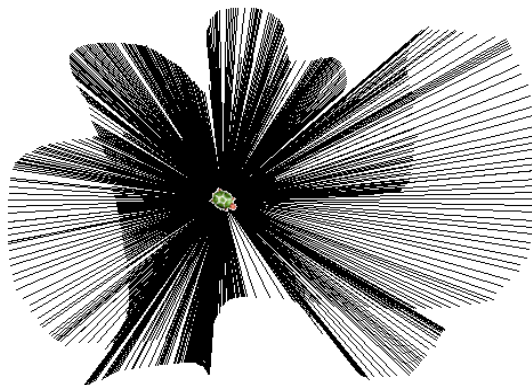
Úloha č.3: Čo to bude robiť, ak vynecháte príkaz `čakaj 50` ?

Úloha č.4: Upravte procedúru `prenasleduj` tak, aby korytnačku myš nepriťahovala ale odpudzovala. (Radšej tam znovu pridajte ten čakací príkaz.)

Úloha č.5: Napíšte procedúru `sleduj` v ktorej sa korytnačka nepohne z miesta, ale bude sa otáčať tým smerom, kde uvidí šípku od myši.

Úloha č.6: Prerobte procedúru z úlohy č.3 na jednoduchý kresliaci program. Procedúra bude bežať, kým sa nestlačí klávesa `Esc`. Pri stlačení `H` sa zdvihne pero, pri stlačení `D` sa pero znovu zapne. Môžete dorobiť ďalšie klávesy na zmenu hrúbky a farby pera.

Úloha č.7(pre machrov): Napíšte procedúru hviezdica, ktorá bude nepretržite kresliť čiary z bodu [0 0] (teda zo stredu) do bodu, v ktorom sa práve nachádza myš. (Pozrite si obrázok.)




Úloha č.8(tiež pre machrov): Prerobte procedúru z úlohy č.4 tak, aby korytnačka utekala od myši rýchlejšie, ak bude myš blízko.

Jedenásta kapitola

Korytnačky

Zle je korytnačke na svete samotnej. A preto programátori, ktorí písali Imagine¹³ dali hlavy dohromady a vymysleli to tak, že nemusí byť korytnačka na svete, stránke či papieri len jediná. Z doterajších kapitol máte približnú predstavu o tom, aký problém je pášť jednu korytnačku. A teraz si predstavte, že ich máme celé stádo. (Nie je isté, že sa korytnačky vyskytujú v stádach, ale stále je to lepšie, ako keby sme tvrdili, že pasieme celý roj korytnačiek.) Skrátka – bude veselo.

Nová korytnačka sa dá vytvoriť jednoducho. Úplne najľahšie je stlačiť tlačidlo . Potom kliknete na plochu a ajhľa – nová korytnačka je na svete. Potom môžete na ňu kliknúť pravým tlačidlom myši a zmeniť jej vlastnosti (napríklad meno, tvar, počítačnú pozíciu, smer a ďalšie veci). Ak potrebujete vytvoriť korytnačku v programe alebo chcete rovno nastaviť nejaké parametre a nechcete sa preklikávať cez okienka, môžete použiť príkaz

nová "korytnačka [meno Benjamín poz [100 100] smer 90 hp 3]

Vznikne nám korytnačka, ktorá sa volá Benjamín, domovskú pozíciu bude mať v bode [100 100], bude otočená na východ a hrúbka jej pera bude 3. Dajú sa nastaviť aj niektoré iné vlastnosti. Ak ich chcete nastaviť, do hranatej zátvorky pridáte ďalšiu dvojicu „vlastnosť hodnota“. Ak chcete, aby mala korytnačka napríklad nejaký iný tvar, pridáte dvojicu tvar martan a v korytnačke, ktorá vznikne by nikto to milé štvornohé zviera nespoznal. Žiadnu vlastnosť samozrejme nastavovať nemusíte. Ak nič nenastavíte, korytnačka dostane štandardné meno (pravdepodobne niečo ako k4) a všetky ostatné parametre budú tiež nastavené na štandardné hodnoty.

Úloha č.1: Príkazom zrušObjekt všetky zrušte všetky korytnačky, ktoré máte a potom vytvorte tri korytnačky, ktoré sa budú volať Hugo, Karolína a Kleofáš, budú mať štandardný tvar a každá bude niekde inde.

Keď teraz zadáte príkaz (napríklad dopredu 100), napodiv sa pohne len jedna korytnačka. (Pravdepodobne Hugo. Záleží od poradia, v akom ste ich vytvárali.) Je to dané tým, že nie všetky korytnačky sú vždy aktívne. Niektoré skrátka spia. Ak chcete zistiť, ktorá korytnačka je hore, zadajte príkaz píš kto. Zoznam kto obsahuje zoznam všetkých korytnačiek, ktoré sú momentálne aktívne.

Ak chceme pohybovať inými korytnačkami, musíme ich najprv zobudiť. Služi na to príkaz odteraz. Ak zadáte príkaz odteraz [Karolína Kleofáš], aktívne budú korytnačky Karolína a Kleofáš. (Hugo nie, ten pôjde spať.) Skúste teraz zopakovať príkaz dopredu 100. Zoznam všetky obsahuje všetky korytnačky. Takže si ich môžeme vypísať príkazom píš všetky, alebo ak chceme, aby všetky korytnačky boli aktívne, môžeme napísať odteraz všetky.

Úloha č.2: Nastavte všetky korytnačky ako aktívne a vyskúšajte si hromadnú drezúru. Čo spraví príkaz nechSmer 90 ? Čo spraví príkaz vpravo náhodne 360 ?

Posledný príkaz je tak trochu prekvapením. Je síce pravda, že všetky korytnačky sa otočili o náhodný uhol, ale všetky o ten istý. Imagine najprv vymyslel náhodný uhol a potom všetky tri korytnačky o ten uhol otočil. Ako to spraví, aby každá korytnačka nabrala iný náhodný smer? Imagine má na to príkaz každá. Ak zadáte príkaz

každá [vpravo náhodne 360]

to, čo je v hranatých zátvorkách sa vykoná pre každú aktívnu korytnačku zvlášť a teda každá korytnačka naberie iný náhodný smer.

¹³ No áno... už autori Comenius Loga...

Niekedy sa stane, že nechceme meniť to, ktoré korytnačky sú aktívne, ale chceme, aby nejakú krátku úlohu vykonali niektoré iné korytnačky. Keď zadáme nasledujúce príkazy

```
odteraz [ Hugo Kleofáš ]  
pre [ Hugo Karolína ] [ dopredu 100 ]
```

Pohnú sa korytnačky Hugo a Karolína bez ohľadu na to, že Karolína práve spí a Kleofáš je hore.

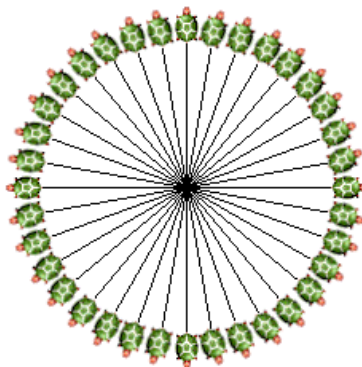
Úloha č.3: Vyskúšajte zadať nasledujúce príkazy:

```
zrušObjekt všetky  
opakuj 50 [ nová "korytnačka [] ]  
každá [ vpravo náhodne 360 ]  
dopredu 150
```

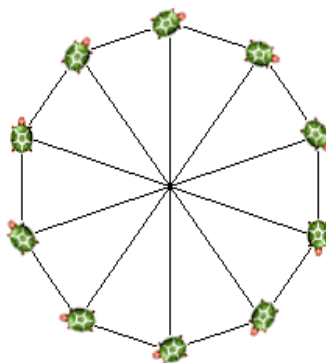
Úloha č.4: Vyskúšajte nasledujúcu sadu príkazov. Prečo to robí, čo to robí? Aký bude stav po druhom príkaze?

```
zrušObjekt všetky  
opakuj 36 [ nová "korytnačka [] odteraz všetky vpravo 10]  
opakuj 360 [ dopredu 2 vpravo 1 ]
```

Úloha č.5: Vytvorte z korytnačiek nasledujúci obrazec:



Úloha č.6: Vytvorte nasledujúci obrazec:



Úloha č.7: Vymyslite ešte niečo pekné.

Dvanásta kapitola

Korytnačka a rekurzia

Kedysi dávno, keď ste chodili do škôlky, určite ste počuli rozprávku o kohútikovi a sliepočke. Pointa bola zhruba v tom, že kohútik bol nenažraný, zabehlo mu zrníčko, išiel sa zadusiť a sliepočka ho musela zachraňovať. Samozrejme bežala ku studni po vodu. Keby studňa nevymýšľala, sliepočka by vodu nabrala, kohútik by sa napil a bolo by po rozprávke – aj keď to by vzhľadom na dĺžku príbehu bol skôr vtip, ako rozprávka. Ale studňa vyhlásila: „Nie, nie, žiadnu vodu nedám, kým nedonesieš od hospodára džbán.“ No a sliepočka ako vzorná manželka bežala za hospodárom. A zasa by to mohlo pekne skončiť, keby hospodár za zapožičanie džbánu nechcel nové traky. A v obchode sliepočke nechceli dať nové traky, kým im nezavedie internet. A poskytovateľ internetových služieb nechcel sliepočke zaviesť internet, kým mu nedonesie originál od Picassa. A Picasso nechcel nič namaľovať, kým nedostane pekingského pinčlíka. A tak to mohlo ísť do nekonečna, keby sliepočka náhodou niekde nenašla zabehnutého pekinéza. Zaniesla ho Picassovi, ten namaľoval originál, provider zaviedol internet, obchodník daroval traky, hospodár požičal džbán, studňa dala vodu, kohútik sa napil a všetko bolo v poriadku.

Podstatnou fázou tej rozprávky je ten moment, keď sliepočka našla toho pinčlíka. Nebyť onej šťastnej udalosti, kohútik by neprežil. V tom bode sa celá rozprávka zvrtila a všetko išlo k lepšiemu.

Podobne to fungovalo, keď sme kedysi dávno v deviatej kapitole počítali faktoriál. Tá funkcia, ktorá ho rátala, vyzerala takto:

```
viem faktoriál :n
  ak2 :n = 1
    [ výsledok 1 ]
    [ výsledok :n * faktoriál (:n - 1) ]
koniec
```

A keď sme chceli vyrátať faktoriál 4, funkcia sa pozrela na svoj tretí riadok a povedala: „Ej, vyrátam ti ja štyri faktoriál, rúči mládenec, či krásna deva, len treba zistiť, čomu sa rovná tri faktoriál. To už len potom vynásobím štvorkou a bude to.“ A tak bolo treba rátať faktoriál 3. Ale druhé volanie funkcie zas vymýšľalo. „Ej, zrátam, zrátam, len by som potrebovala vedieť, čomu sa dva faktoriál rovná. Ten vynásobím trojkou a hotovo.“ A tak bolo treba rátať faktoriál 2. Funkcia sa zavolala tretíkrát a zase hovorí „Vyrátať dva faktoriál je hračka. Vynásobím dvojkou jedna faktoriál. Koľko je jedna faktoriál?“ A tak sa funkcia volala štvrtýkrát. No a v tomto momente došlo k šťastnému obratu rovnajúcemu sa nálezu pinčlíka. Funkcia prestala vymýšľať, keď jej dali vyrátať faktoriál 1, povedala, že jedna a všetko sa to začalo vracáť naspäť. Tretie volanie funkcie (ktoré rátalo faktoriál 2) vynásobilo jednotku dvomi a vrátilo výsledok 2, druhé volanie funkcie (rátalo faktoriál 3) vynásobilo dvojkou tromi a vrátilo výsledok 6 a prvé volanie funkcie tú šestku vynásobilo štyrmi a hrdo vyhlásilo, že „štyri faktoriál je dvadsať štyri“. No a takto nejak funguje rekurzia.

Rekurzia sa dá využiť aj na generovanie niektorých zaujímavých obrázkov. Urobme si napríklad procedúru čiara, ktorá bude mať dva parametre. Prvý bude dĺžka a druhý úroveň rekurzie. Ak bude úroveň rekurzie 0, nakreslí sa iba čiara danej dĺžky. Takže čiara 300 0 nakreslí:



Pri vyššej úrovni rekurzie sa ale procedúra bude správať rafinovane. Zavolá opäť procedúru čiara,

ale s tretinovou dĺžkou a úroveňou o 1 menšou. Potom sa otočí vľavo o 60 stupňov, znovu zavolá procedúru čiara s úroveňou o 1 nižšou, otočí sa vpravo o 120 stupňov, znovu zavolá procedúru čiara a nakoniec sa otočí vľavo o 60 stupňov a zas zavolá procedúru čiara.

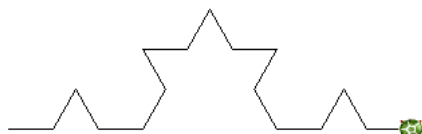
Kód procedúry bude teda vyzerat takto:

```
viem čiara :dĺžka :úroveň
  ak2 :úroveň = 0
    [dopredu :dĺžka]
  [
    čiara (:dĺžka / 3) (:úroveň - 1)
    vľavo 60
    čiara (:dĺžka / 3) (:úroveň - 1)
    vpravo 120
    čiara (:dĺžka / 3) (:úroveň - 1)
    vľavo 60
    čiara (:dĺžka / 3) (:úroveň - 1)
  ]
koniec
```

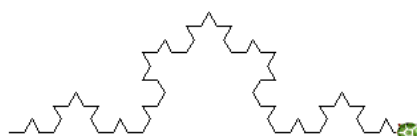
Ak zadáme príkaz čiara 300 1, korytnačka nakreslí



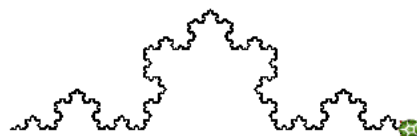
ak zadáme príkaz čiara 300 2, korytnačka nakreslí



ak zadáme príkaz čiara 300 3, korytnačka nakreslí



a ak zadáme príkaz čiara 300 7, korytnačka nakreslí



Ak ste niekedy počuli o fraktáloch, tak takto vzniká jeden z ich typov.

Úloha č.1: Vyskúšajte si funkciu čiara.

Úloha č.2: Zistite, čo urobí funkcia

```
viem pokus :n  
  ak2 :n = 0  
    [ ukonči ]  
    [ pokus (:n - 1) do :n vpravo 30 ]  
koniec
```

Úloha č.2: Zistite, čo vypočíta funkcia

```
viem čárymáry :n  
  ak2 :n = 1  
    [ výsledok 1 ]  
    [ výsledok :n + čárymáry (:n - 1) ]  
koniec
```


Trinásta kapitola

Korytnačka a zoznamy

Táto kapitola bude ťažká pre ťažká, pretože sa v nej bude málo kresliť a veľa rozmýšľať. Ide však o vec veľmi dôležitú, bez ktorej by mnohé veci v Imagine nefungovali – ide o zoznamy.

So zoznamami sme sa už niekoľkokrát stretli bez toho, že by sme na ne nejako zvlášť upozorňovali. Ak sme napríklad potrebovali určiť pozíciu nejakej korytnačky na ploche, slúžil na to príkaz `poz`. Ten nám ako výsledok nevrátil jediné číslo, ale zoznam dvoch čísel, ktoré reprezentovali x-ovú a y-ovú súradnicu korytnačky. Príkaz `všetky` nám vrátil zoznam všetkých korytnačiek a aj keď sme zadali príkaz `piš "Žinčica`, nechali sme iba vypisovať zoznam obsahujúci písmenká `Ž, i, n, č, i, c, a a`.

Do zoznamov je možné ukladať všeličo. Obrázky, čísla, písmená, slová aj iné zoznamy. Zoznamy môžeme s pomocou špeciálnych funkcií prezerať a meniť. Niektoré funkcie chcú zoznamy ako vstupné parametre a niektoré funkcie ich zase vrátia ako výsledok. Skrátka, kam sa pozriete, všade samý zoznam. A úlohou tejto kapitoly je naučiť vás so zoznamami pracovať.

Pre začiatok si nejaký zoznam uložíme do premennej:

```
urob "zoznamček [ a be ce de e ef ge ha ]
```

Prvý prvok zoznamu nám vyberie funkcia `prvý`. Takže ak napíšete príkaz `piš prvý :zoznamček`, Imagine vám napíše `a`. Posledný prvok zoznamu nájde funkcia `posledný` – hádajte ktorá :-) – no áno, je to funkcia `posledný`. Takže príkaz `piš posledný :zoznamček` nám napíše `ha`. Ak chceme vypísať tretí prvok zoznamu, môžeme použiť príkaz `piš prvok 3 :zoznamček`. (Napíše to `ce`.)

Príkaz `bezPrvého` usekne zo zoznamu prvý prvok, takže príkaz

```
zobraz bezPrvého :zoznamček
```

vypíše `[be ce de e ef ge ha]`.¹⁴ Podobne príkaz `bezPosledného` odsekne zo zoznamu posledný prvok. Príkaz `bezPozície` zo zoznamu vynechá prvok na určenej pozícii. Teda napríklad príkaz

```
zobraz bezPozície 3 :zoznamček
```

vypíše `[a be de e ef ge ha]`.

Ak chceme dva zoznamy spojiť, zluží na to príkaz `veta`. Takže príkaz

```
zobraz veta :zoznamček [ jeden dva tri ]
```

vypíše `[a be ce de e ef ge ha jeden dva tri]`

Ak chceme vedieť počet prvkov zoznamu, slúži na to (prekvapivo) funkcia `počet`. (Takže `piš počet :zoznamček` napíše `8`.)

Dobre. Máme milión nových funkcií a teraz ako ich použiť. Predstavte si, že máme v premennej bodíky uložený nasledujúci zoznam bodov:

```
[ [0 0] [20 0] [20 20] [0 20] [0 40] [-20 40] [-20 20]  
[-40 20] [-40 0] [-20 0] [-20 -20] [0 -20] [0 0] ]
```

(všimnite si, že je to zoznam malých dvojprvkových zoznamov). Chceli by sme, aby ich korytnačka všetky postupne navštívila – aby sa pre každý bod zavolała rocedúra `nechPoz`, ktorá korytnačku na daný bodík premiestni. Použijeme cyklus `opakuaj` a opakovať budeme toľkokrát, koľko bodov máme v zozname. Pri prvom opakovaní treba vybrať prvý prvok zoznamu, pri druhom druhý... Na

¹⁴ Príkaz `zobraz`, skrátka `zo` na rozdiel od príkazu `piš` nakreslí okolo vypisovaného zoznamu aj jeho hranaté zátvorky.

to, aby sme zakaždým vybrali správny bodík, použijeme funkciu **počítadlo**, ktorá nám počíta, koľkýkrát už cyklus **opakuj** beží. Takže vykreslenie bodíkov bude vyzeráť nasledovne:

```
opakuj počet :bodíky [ nechPoz prvok počítadlo :bodíky ]
```

Úloha č.1: Urobte si premennú **bodíky**, ktorá obsahuje uvedený zoznam a vyskúšajte si to. Pozerajte sa na uvedený príkaz dovedy, kým nepochopíte, ako vlastne funguje. (Prípadne sa spýtajte.)

Ešte jedna ukážka práce so zoznamami. Potrebujeme napísať funkciu **reverz**¹⁵, ktorá dostane na vstupe zoznam a vráti zoznam otočený naopak. Teda ak na vstupe dostane [a b c], ako výsledok vráti [c b a]. Použijeme v nej našu milú rekurziu. Funkcia funguje jednoducho. Ak má zoznam dĺžku 1, netreba nič obracať. A ak má väčšiu dĺžku, odtrhneme prvý prvok, na zvyšok zoznamu poštveme pôvodnú otáčaciu funkciu (a budeme dúfať, že sa časom dopracuje k jednoprvkovému zoznamu, kde sa situácia otočí) a prvý prvok priplácneme na koniec:

```
viem reverz :a
  ak počet :a <= 1 [ výsledok :a ]
  výsledok (veta reverz bezPrvého :a prvý :a)
koniec
```

Predstavte si, že funkciu **reverz** predhodíte zoznam [a b c d]. Keďže má štyri prvky, **reverz** má vrátiť výsledok (veta **reverz** [b c d] a). Na to sa ale musí funkcia **reverz** znovu zavolať, aby zistila **reverz** [b c d]. Toto druhé volanie znovu utrhne prvý prvok a nechá si zistiť, čo je **reverz** [c d]. Tretie volanie zas odtrhne prvý prvok, nechá si zistiť **reverz** [d], konečne sa dozvie, že [d], (ak má zoznam jeden prvok, tak ho otočíť vieme a to je naša výhra – spomeňte na rozprávku z predošlej kapitoly) priplácneme c zaňho a vráti výsledok [d c]. Druhé volanie sa teda dozvie, že **reverz** [c d] je [d c], prilepí za neho b a vráti [d c b]. No a prvé volanie, keďže sa už dozvedelo, čo je to **reverz** [b c d], víťazoslávne prilepí na koniec to a a vráti výsledok [d c b a].

Úloha č.2: Vyskúšajte a pochopte.

Úloha č.3: Napíšte funkciu **ptp**, ktorá na vstupe dostane zoznam a vypíše jeho prvý, tretí a posledný prvok.

Úloha č.4: Napíšte funkciu, ktorá dostane na vstupe zoznam a ako výsledok vráti ten istý zoznam bez posledných troch prvkov.

Úloha č.5: Napíšte funkciu **plus**, ktorá dostane na vstupe zoznam čísel a vypíše čísla o jedna väčšie, než sú tie v zozname.

Úloha č.6: Napíšte funkciu **xx**, ktorá na vstupe dostane zoznam bodov (môžete použiť **bodíky**) a vypíše ich x-ové súradnice.

Úloha č.7: Napíšte funkciu **max**, ktorá dostane na vstupe zoznam čísel ako výsledok vráti najväčšie z nich.

15 Túto funkciu robíme len z cvičných dôvodov. Imagine má funkciu **prevráť**, ktorá robí presne to isté.

Štrnásť kapitola

Korytnačka mnohých tvári

Keď ste skúšali nejaké veci s Imaginom, je veľmi pravdepodobné, že sa vám podarilo zmeniť tvar korytnačky. Ak sa vám to ešte nepodarilo, stačí na korytnačku kliknúť pravým tlačidlom myši, z ponuky vybrať Zmeň k1 (samozrejme, ak sa korytnačka nevolá k1, bude tam niečo troška iné) a v dialógovom okne, ktoré na vás vyskočí zvolíte gombík Tvar... a môžete si vybrať, či má korytnačka vyzerať, ako robot, domorodec, asteroid alebo Nemecko. Tieto a iné korytnačky sú súčasťou Imaginu a môžete ich vo svojich projektoch využiť. Čo ale robíte vtedy, ak nemáte k dispozícii korytnačku, akú práve potrebujete? Ako vždy máte dve možnosti – prvá je podľaŕhnúť beznádeji a na všetko sa vykašľať. Druhá je urobiť si korytnačku akú potrebujete. Nástroj na vytváranie nových korytnačiek sa volá *LogoMotion*.



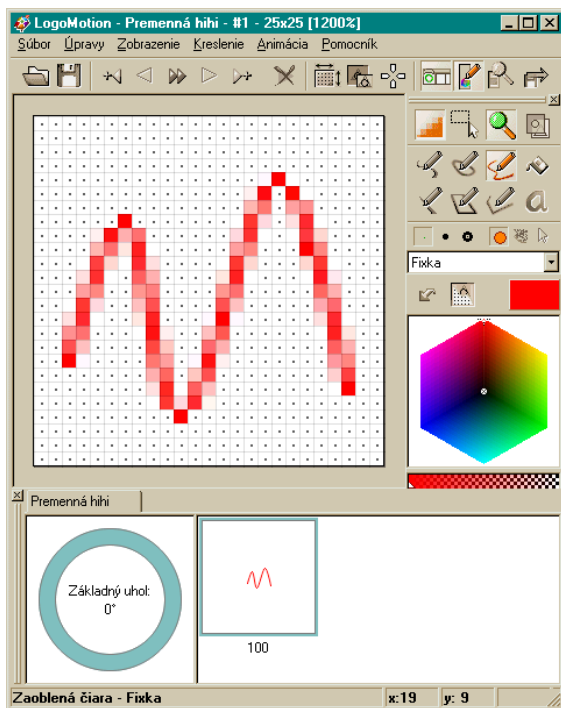
A opäť máte dve možnosti. Prvá je, že spustíte LogoMotion priamo z pracovnej plochy, urobíte si nový tvar korytnačky a keď ste hotoví uložíte si ju (v menu Súbor a Uložiť ako...) k ostatným korytnačkám a načítate s pomocou menu alebo s pomocou príkazu nechTvar. Ak napríklad chceme, aby korytnačka k2 získala tvar uložený v súbore *robot.lgf* z adresára *Imagine\Obrázky*, použijeme príkaz `k2'nechTvar "robot`. Tento prístup má tú výhodu, že korytnačku, ktorú vytvoríte, môžete používať vo viacerých projektoch a môžete ju dať k dispozícii komu chcete. Nevýhodou je, že korytnačku musíte skladovať v adresári *Imagine\Obrázky* (alebo niekde inde a potom v mene korytnačky musíte uviesť celú cestu) a ak projekt ktorý tú korytnačku používa presuniete na počítač, kde ju nemáte, alebo máte uloženú inde, tak nebude fungovať.

Druhá možnosť je urobiť si obrázkovú premennú. Môžete to spraviť napríklad príkazom

```
urob "hihi PrázdnnyObrázok
```

V okne **pamäť** sa vám objaví premenná hihi, ktorá je obrázková a keď na ňu kliknete a skúsíte zmeniť jej hodnotu, otvorí sa vám *LogoMotion* a vy sa môžete umelecky realizovať. Keď dokreslíte, vyberiete z menu *Súbor/Skončiť* a keď sa vás program spýta, či chcete uložiť zmeny, kliknete, že áno.¹⁶ Nový obrázok sa uloží do premennej a tú môžete používať bežným spôsobom (napríklad `k2'nechTvar :hihi`). Výhodou tohto prístupu je, že obrázok je priamo súčasťou projektu. Ak ho ale chcete dať niekomu inému, musíte ho znovu otvoriť v *LogoMotion* a uložiť do súboru.



Fajn. Takže vieme, čo chceme urobiť, máme spustený LogoMotion, jediná otázka je, ako to urobiť. Ak kreslíte iba jeden obrázok, situácia je celkom jednoduchá. Nástroje na kreslenie a výber farby sú podobné, ako pri iných kresliacich programoch. Štvorčeky s bodkou sú priesvitné. Ak chcete niečo, čo už ste zafarbili spriesvitniť, miesto fixky si nastavte gumu. Jednotlivé kresliace nástroje tu do detailov rozoberať nebudeme. Vyskúšajte si to sami. Celé je to vymyslené pekne a intuitívne a ak chce

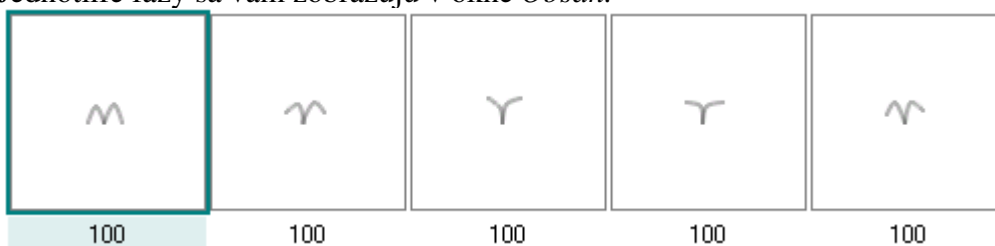



16 Celá táto akcia sa dá skrátiť na stlačenie tlačidla **F12**, ktoré znamená „uložiť a skončiť“

človek spraviť nejakú špeciálnu korytnačku, môže sa s tým hrať dlhé hodiny.

Ako ale viete, korytnačka, to nie je len jeden obrázok. Je tam mnoho ďalších detailov. Niektoré korytnačky sa hýbu. Niektoré korytnačky menia tvar podľa toho, ktorým smerom sú otočené. (Áno, aj úplne najštandardnejšia korytnačka má túto vlastnosť – keď je otočená vpravo, použije sa iný obrázok, než keď je otočená hore. Podľa toho vlastne poznáme, ktorým smerom pôjde.) Korytnačke treba určiť, ktorý je jej základný bod, ktorým píše atď. A LogoMotion má nástroje, s ktorých pomocou môžete toto všetko urobiť.

Povedzme si najprv niečo o animáciách. Každá animácia sa skladá z jednotlivých fáz. Ak chcete, aby sa vám korytnačka hýbala, treba nakresliť niekoľko obrázkov a tie sa budú na mieste korytnačky cyklicky vykresľovať. Ak chcete korytnačke pridať ďalšiu fázu, slúži na to tlačidlo . Objaví sa vám úplne nový obrázok a do neho môžete ďalšiu fázu nakresliť. Aby ste sa s novou fázou trafili na presne to miesto, na ktoré sa trafiť potrebujete, môže byť užitočné zapnúť si priesvitky (tlačidlo ). Vtedy sa vám predošlá fáza slabúčko nakreslí pod plochu, aby ste vedeli, kde čo je. Jednotlivé fázy sa vám zobrazujú v okne *Obsah*.




Ak si chcete animáciu pustiť už v LogoMotion-e, použijete tlačidlo . Ak takýto animovaný obrázok použijete ako tvar korytnačky, korytnačka sa vám bude sama hýbať.

Takto vytvorená korytnačka má ale tú vlastnosť, že vyzerá stále rovnako bez ohľadu na to, ktorým smerom je otočená. Ak sa jedná o jednoduchú animáciu letiacej vrany, je to v poriadku. Ale ak by sa jednalo o „pracovnú“ korytnačku, ktorá mimochodom vôbec nemusí byť animačná, je dôležité, aby sa z jej vzhl'adu dalo určiť, ktorým smerom je otočená.

Na to, aby sa korytnačka menila v závislosti od toho, ktorým smerom je otočená, treba spraviť viacero záberov. Nový záber pridáte tak, že kliknete pravým tlačidlom myši na koliesko v obsahu (obrázok vpravo) a vyberiete *Úpravy/Vložiť nový za*. Koliesko sa rozdelí na dve rovnaké časti, medzi ktorými sa môžete prepínať kliknutím myši. Ak bude korytnačka otočená do hornej polovice, bude sa zobrazovať prvý záber. Ak bude otočená do dolnej polovice, bude sa ukazovať druhý záber. (Oba tieto zábery samozrejme môžu byť animované.)



Záberov môžeme vložiť viacero. Kruh reprezentujúci jednotlivé smery sa potom rozdelí na viacero rovnakých častí. Každý záber bude vykresľovaný iba vtedy, keď bude korytnačka otočená „tým jeho“ smerom. (Vyznačený záber na obrázku vľavo bude vykreslený, ak bude azimut korytnačky medzi 30° a 90°. Korytnačka z tohto obrázku má 6 záberov.) Štandardná korytnačka obsahuje až 24 záberov. (Z toho sa dá ľahko vypočítať, že ak ju budete otáčať pomaly niektorým smerom, zmenu smeru zaregistrujete iba každých 15 stupňov.)

Ak chcete korytnačke nastaviť základný bod, použijete tlačidlo . Na korytnačke sa objaví zameriavací kríž, z ktorého pomocou môžete posunúť základný bod tam, kde ho potrebujete mať.

LogoMotion je šikovný a relatívne rozsiahly nástroj na podrobný popis ktorého zďaleka nestačí jedna kapitola. Je tu veľa priestoru na skúšanie. Takže skúšajte a ak niečo neviete, obráťte sa na manuál, na vyučujúceho či iného odborníka alebo na lampáreň.

Úloha č.1: Vytvorte nejakú skvelú korytnačku.


Pätnästa kapitola

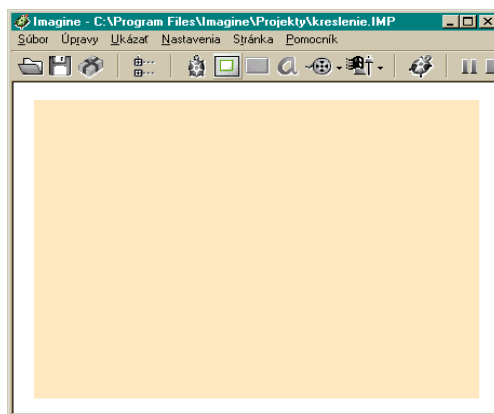
Korytnačka a umenie

Naše rozprávanie o korytnačkách a Imagine na nejaký čas zmení podobu. Doteraz sme si väčšinou rozprávali o nových príkazoch a o tom, ako ich využiť. Teraz to budeme robiť trochu inak. Na začiatku bude vždy stáť nejaký projekt. Budeme mať nejakú väčšiu vec, ktorú budeme chcieť urobiť a budeme si ukazovať, ako na to. Kým to bude hotové, môže často prejsť niekoľko kapitol. A popri tom sa naučíte ako veci fungujú a ako sa dajú šikovne urobiť.


Pripravte sa, prichádza prvý projekt – **kreslenie**. Pokúsime sa spraviť program, ktorý robí niečo podobné, ako kreslítko od Microsoftu (alebo ľubovoľné iné kreslítko). Malo by to vedieť meniť farbičky, malo by byť vidieť, ktorá farba je zapnutá, malo by to mať gumu, malo by to vedieť kresliť rovné čiarky, obdĺžniky a kolieska, vypĺňať uzavreté oblasti, písať do obrázku písmenká a potom to celé vedieť uložiť na disk a znovu prečítať.

Každému, kto aspoň matne tuší, koľko roboty nás čaká, sa pri prečítaní predošlých riadkov museli zachvieť kolená. Ale nebojte sa. Nie je to otázka jednej kapitoly, ani jedného dňa. Začneme pomaly a postupne budeme vylepšovať.

Takže najprv zrušme korytnačku. (príkaz `zrušObjekt všetky`) Potom si na plochu pridáme nový papier. Spravíme to tak, že klikneme na tlačidlo  a potom na ploche potiahneme myšou, kde papier chceme mať. Taký papier je praktická vec. Je to niečo ako obrazovka v obrazovke. Papier môže mať svoje vlastné korytnačky, ktoré z neho nemôžu zliezť. Keď potrebujeme nejakú časť obrazovky vyčleniť na nejaký účel, je dobré si na to zriadiť papier. Náš papier bude presne tá časť obrazovky, do ktorej sa bude kresliť a zvyšok obrazovky si necháme na ovládacie prvky. Ak úspešne pridáte papier, malo by vám to vyzerať podobne, ako na obrázku vpravo.



Teraz budeme chcieť zabezpečiť, aby sa na ten papier dalo kresliť. Jednu z možností, ako sa to dá urobiť, sme už ukázali v 6. kapitole. Teraz si ukážeme inú možnosť.

Najprv si na papier pridáme korytnačku (napríklad tak, že klikneme na tlačidlo  a potom na papier). Korytnačke zmeníme meno na **Písatko** (klikneme na ňu pravým tlačidlom, zvolíme *Zmeň kl* a nové meno napíšeme do kolónky *Meno*.) Na korytnačke nás momentálne zaujíma najmä tá jej vlastnosť, že vie po papieri písať.

Pokúsime sa dosiahnuť, aby sa vždy, keď na papier vstúpime myšou, korytnačka držala presne pod myšou. Pomôžu nám starí známi – udalosti. Udalosti sa totiž nemusia vzťahovať len na korytnačky. Aj taký papier vie svoje udalosti spracovávať. Preto sa prepnite do okna *pamäť*, dvojklíknite na váš papier (pravdepodobne sa bude volať `papier1`), zapnite si udalosti, pridajte udalosť `priPohybeMyši` a do patričnej priehradky Imaginu napíšte, aby vtedy spravil

`Písatko'nechPoz pozMyši`

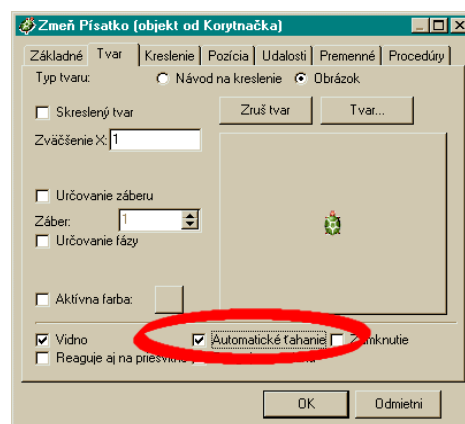
Skúste teraz prejsť myšou ponad papier a uvidíte, ako sa bude správať korytnačka **Písatko**.

Úloha č.1: Spravte to všetko až potiaľto. Dajte si pozor, aby ste pridali udalosť papiera a nie korytnačke, alebo stránke.

Dobre. Korytnačka sa hýbe spolu s myšou a všade kreslí. Chcelo by to zariadiť, aby kreslila iba vtedy, keď máme stlačené ľavé tlačidlo na myši. To sa dá zariadiť pomerne jednoducho. Korytnačke

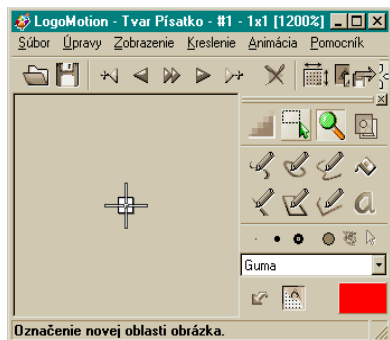
Písatko pridáme dve udalosti. Povieme jej, aby priĽavomDolu spravila peroDolu a priĽavomHore spravila peroHore. (Pozor, toto musia byť udalosti korytnačky a nie udalosti papiera. Korytnačka totiž stále trčí pod myšou a na papier samotný momentálne nemáte ako kliknúť.) Vyskúšajte si to.



Ak ste si to vyskúšali, vidíte, že je na svete problém. Korytnačka síce prestala čmárať, ale keď ste na ňu klikli, odmietla sa pohnúť z miesta dovtedy, kým tlačidlo na myši znovu nespustíte. Stlačenie myši totiž nezaznamenal papier, ale samotná korytnačka. Tá si myš uzurpovala pre seba a odmieta ju uvoľniť až dovtedy, kým to nešťastné tlačidlo nepustíte. Túto nepríjemnosť môžeme ale celkom jednoducho odstrániť. Kliknite pravým tlačidlom na korytnačku, vyberte *Zmeň Písatko* a v záložke *Tvar* zaškrtnite možnosť *Automatické ťahanie*. To spôsobí, že keď kliknete na korytnačku, budete ju ťahať so sebou až dovtedy, kým tlačidlo na myši nepustíte, čo je zhodou okolností presne to, čo chceme dosiahnuť.



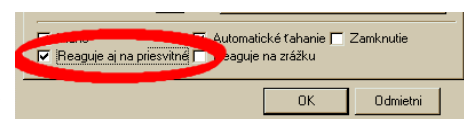
Úloha č.2: Vyskúšajte.

Fajn. Robí nám to, čo potrebujeme a dokonca k tomu nepotrebujeme spustiť žiaden príkaz. Už to má len jednu drobnú slabinu – zavadzia tam tá korytnačka. Skryť ju nemôžeme. To by sa na ňu nedalo kliknúť a nemohla by spustiť pero na písanie. A vláčiť ju stále pod myšou – to nevyzerá pekne.



Ale aj to sa dá uhrať. Najprv zmeníme tvar korytnačky (klik pravým tlačidlom...*Zmeň Písatko*... a potom kliknúť na to tlačidlo s obrázkom korytnačky.) Tvar zmeníme pomerne dramaticky. Najprv zmeníme rozmery korytnačky na 1x1 štvorček (v LogoMotion tlačidlo ) a potom už si domyslíte) a potom ten štvorček vygumujeme (nezabudnite si miesto fixky zapnúť gumu). Nakoniec ešte treba nastaviť základný bod korytnačky na ten jediný štvorček, ktorý jej zostal (tlačidlo ) takže nakoniec by to malo vyzeráť ako na obrázku vľavo. Keďže má korytnačka len jeden štvorček a aj ten je priesvitný, je v podstate neviditeľná.

Posledným problémom je, že ak korytnačke kliknete na jej priesvitné časti, bežne na to nereaguje. Takže jej treba povedať, aby toto správanie zmenila a brala svoj jediný štvorček vážne, aj keď ho nevidieť. Klik pravým tlačidlom teraz nezaberie (z práve uvedených dôvodov) takže treba na *Písatko* dvojkliknúť v okne *Pamäť*, vybrať záložku *Tvar* a zaškrtnúť *Reaguje aj na priesvitné*. Od tejto chvíle začne jednobodová priesvitná korytnačka znova vnímať naše klikanie myšou a bude robiť presne to, čo od nej potrebujeme.



Úloha č.3: Urobte projekt až potiaľto, vyskúšajte a uložte si ho na disk. Nabudúce v ňom budeme pokračovať !!!

Na záver jeden teoretický postreh a jedna drobná praktická rada. Najprv k tej teórii. V 6. kapitole sme naznačili, ako môže vzniknúť kresliaci program. A s fintami z kapitoly 10 by sme vedeli vytvoriť príkaz, ktorý by nám zabezpečil podobnú funkčnosť ako to, čo sme tu práve spravili. Teraz sme ale nevyrobili príkaz. Namiesto toho sme na plochu rozhodli nejaké objekty a povedali im, ako sa majú správať. K tomu je síce užitočné vedieť vytvárať aj nejaké zložitejšie príkazy a túto zručnosť budeme potrebovať, ale nadreli sme sa podstatne menej, než keby sme to

mali zveriť jednému príkazu a pri tomto prístupe sa nám projekt bude oveľa jednoduchšie rozširovať.

A tá drobná praktická rada? Ak chcete zmazať tie čmáranice, čo ste na papier vyprodukovali, napíšte papier1'zmaž.

Pokračovanie nabudúce :-)

Šestnásta kapitola



Korytnačka a tlačidlá

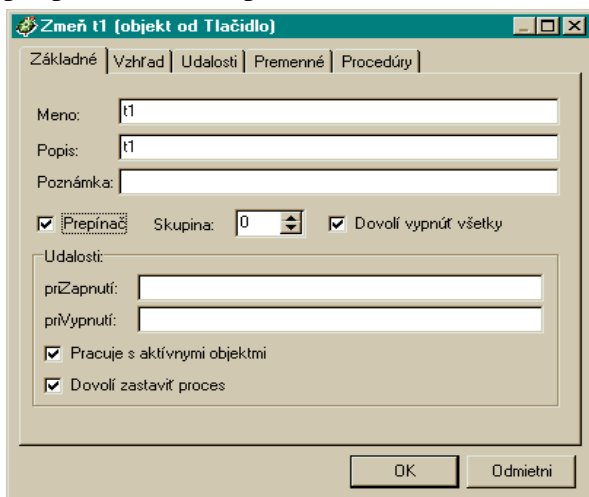
Naše rozprávanie o korytnačke Žofke sa dostalo do okrúhlej šestnásťtej kapitoly. (Áno okrúhlej – v dvojrovej sústave, ktorú používajú počítače, sa 16 zapisuje ako 10000 a v šestnásťrovej sústave 0x10). Budeme v nej vylepšovať to, čo sme vytvorili v 15. kapitole a pridáme niektoré novinky.

Tlačidlá sú bežnou súčasťou nášho života. Či už ide o tlačidlá hardwarové (vypínače, zvončeky, tlačidlá na videu, štartovátka na raketu, klikátka na myši,...) alebo tlačidlá softwarové (rôzne gombíky, súčasti nástrojových líšt, hypertextové odkazy,...) Tie softwarové sú niekedy vyčakané, niekedy jednoduché a obsahuje ich každý lepší program. Sme na ne zvyknutí, stláčame ich a očakávame, že sa pri tom niečo udeje.


A ani Imagine sa nevyhol možnosti ozdobiť váš projekt rôznymi gombíkmi. Skúste kdesi v hĺbke harddisku vyhľadať projekt, ktorý sme začali v minulej kapitole a nahrajte si ho. (Kto ho nemá uložený, ten môže buď horko zaplakať, alebo poprosiť kolegu, či sa nad ním nezmluje, alebo si ho celý rýchlo urobiť ešte raz.) Môžete si skontrolovať, či neviditeľná korytnačka ešte stále žije a či je ochotná stále pre vás kresliť. Ak všetko funguje tak, ako má, môžeme začať s pridávaním tlačidiel.

Iste ste si všimli, že ak je niečoho veľa, tak to nie je vždy na osoh. A zvlášť to platí o množstve čmáraníc na papieri. V minulej kapitole sme spomenuli spôsob, ako papier zmazať, ale klepať zakaždým príkaz do príkazového riadku nemusí vyhovovať každému. Preto si teraz skúsime urobiť mazávacie tlačidlo.

Nové tlačidlo pridáme na plochu tak, že klikneme na tlačidlo  a potom klikneme tam, kde naše nové tlačidielko chceme mať. Na ploche sa nám objaví niečo podobné tomuto:  Na prvý pohľad žiadna veľká výhra. Nevyzerá to nijak úžasne, dá sa na to síce klikať, ale nič to nerobí. Našťastie nič nie je stratené a novovzniknuté tlačidlo je úplne v našej moci a môžeme si ho prispôbiť naším potrebám.



Takže najprv výzor. Tlačidlo môže byť textové, obrázkové alebo aj také, aj také. Podobne, ako keď sa mení korytnačka, aj na tlačidlo treba kliknúť pravým tlačidlom myši a vybrať možnosť *Zmeň*. Objaví sa dialógové okno, aké môžete vidieť na obrázku vľavo (možno o čosi chudobnejšie, niektoré riadky sa objavujú iba za istých okolností). Môžete zmeniť meno tlačidla¹⁷, popis (to je to, čo je na tlačidle napísané) a iné veci, o ktorých sa zmienime o chvíľu. Ak kliknete na roletku *vzhľad* a potom na najväčšie tlačidlo, ktoré tam nájdete, spustíte LogoMotion a v ňom môžete nakresliť tvar, ktorý chcete mať na tlačidle zobrazený. Odporúčame použiť na obrázky na tlačidlá rozmer 16 × 16 (malá ikona).


Tiež tu môžete upraviť veľkosť tlačidla, aby sa vám tam ten obrázok, ktorý ste nakreslili vôbec zmestil. Ak tam necháte obrázok aj popis, môže vaše tlačidlo vyzeráť napríklad takto: 

Tlačidlo už máme, má sa celkom k svetlu, aj klikať sa naňho dá, ale stále to má jednu slabinu – keď ho stlačíte, vôbec, ale vôbec nič sa neudeje. Poďme teda zariadiť, aby prítomnosť tlačidla na ploche mala nejaký zmysel. Znovu vbehnite do dialógu, v ktorom meníte tlačidlo a do

¹⁷ Meno tlačidla je niečo podobné ako meno korytnačky. S jeho pomocou sa na tlačidlo môžete odvolávať zvnútra programu.

riadku *priZapnutí* napíšete príkaz, ktorý sa má vykonať, keď tlačidlo stlačíte: **papier1'zmaž**. (Poznámka – keby ste tam napísali iba **zmaž**, mazala by sa plocha (na ktorej nie je nič nakreslené) a papier by ostal nezmazaný. Ak chceme zavolať procedúru **zmaž**, ktorá sa týka nášho papiera, treba napísať meno toho papiera, tesne za ním apostrof – to je tá jedna úvodzovka hore – a tesne za apostrofom príkaz, ktorý má daný objekt vykonať.) No a to je všetko. Keď teraz kliknete na tlačidlo, stane sa presne to, čo ste chceli dosiahnuť – papier sa zmaže.

Úloha č.1: Dorobte zmazávacie tlačidlo, vyrobte mu obrázok, ktorý pôsobí profesionálnym dojmom a sprevádzkujte ho.

Teraz by to chcelo nejaké farby. Na začiatok si spravíme jedno pastelkové tlačidlo, ktoré by mohlo mať rovnaké rozmery, ako zmazávacie, meno **biela** a mohlo by vyzeráť približne takto:  Dbajte na to, aby v kolónke *Popis* nebol žiaden text, ktorý by tentokrát na tlačidle iba zavádzal. Spomenieme si, že naša neviditeľná korytnačka sa volala *písatko* a do kolónky *priZapnutí* napíšeme *písatko'nechFp "biela* Kliknutie na toto tlačidlo spôsobí, že korytnačka bude kresliť na bielo. (Vyskúšajte si to.)

Máme tlačidlo, s pomocou ktorého vieme prepnúť farbu na bielu. Problém ale je, že nemáme žiadne iné prepínacie tlačidlo. Takže máme tri možnosti. Buď nadosmrti kresliť bielou, alebo si na iné farby prepínať z príkazového riadku, alebo si dorobiť ďalšie tlačidlá. Áno, cé je správne, dorobíme si ďalšie tlačidlá.

Najjednoduchšie to spravíme tak, že klikneme na bielu ceruzku pravým tlačidlom myši, zvolíme možnosť *Skopíruj do schránky* a potom klikneme pravým tlačidlom myši tam, kde chceme mať nové tlačidlo a zvolíme možnosť *Prilep zo schránky*. Tlačidlu upravíme polohu, v LogoMotione ho prefarbíme na žltú, zmeníme meno na **žltá** a v kolónke *priZapnutí* zmeníme "biela na "žltá". A môžeme sa tešiť, že máme už dve farby.

Teraz môžeme prepínať medzi bielou a žltou, ale z tlačidiel nijak nevieme uhádnuť, čo je práve zapnuté. Bolo by oveľa lepšie, keby to tlačidlo, ktoré sme práve stlačili, ostalo stlačené až

dovtedy, kým nezvolíme nejakú inú farbu. Aj to sa dá dosiahnuť celkom jednoducho. Najprv musíme oboj tlačidlám nastaviť, že sú to prepínače (tak ako na obrázku vľavo). Prepínač je tlačidlo, ktoré môže ostať zapnuté. Keď to zapnete, objavia sa vám nejaké ďalšie nastavenia ohľadom skupiny, do ktorej prepínač patrí. Ako číslo skupiny je štandardne nastavená nula. Vtedy je prepínač „sám pre seba“ – keď na neho kliknete, zapne sa, keď kliknete znovu vypne sa. Keď ho ale do nejakej skupiny zaradíte, spôsobí to tú zmenu, že keď ho zapnete, všetky ostatné prepínače z tej skupiny sa vypnú. Keď ešte navyše zrušíte možnosť *Dovolí vypnúť všetky*, bude vždy aspoň jedna pastelka zapnutá. A to je presne to, čo potrebujeme.

Úloha č.2: Vyskúšajte si to, dorobte ešte červenú, zelenú, modrú a čiernu pastelku a zabezpečte, aby bolo vidieť, ktorá pastelka je práve zapnutá.

Úloha č.3: Pridajte dve tlačidlá, ktorými sa dá prepínať medzi hrubým a tenkým perom.

Po splnení oboch úloh by vaša pracovná plocha mohla vyzeráť podobne ako na obrázku dole. Náš kresliaci program začína pomaly skutočne vyzeráť ako kresliaci program.



Sedemnásť kapitola


Korytnačka a posúvatka

Prvá vec, ktorú na našom kreslítku ešte treba zlepšiť, je výber farieb. Zatiaľ ich tam máme šesť. To je lepšie, ako jedna, ale na nejaké veľké umenie to stále nevyzerá. Chcelo by to mať možnosť vybrať si ktorúkoľvek z farieb, ktoré môže počítač zobraziť.

Na takýto výber farieb má Imagine pripravený dialóg. Skúste v príkazovom riadku zadať príkaz

píš pomôckaFarba 0

Príkaz `pomôckaFarba 0` vám vyrobí okienko, v ktorom si môžete vybrať niektorú z prednastavených farieb, alebo si s pomocou červenej, zelenej a modrej zložky namiešať takú farbu, aká vám vyhovuje. Tá nula príkazu povie, akú farbu má nastaviť na začiatok (nula je čierna). Výsledok príkazu (buď meno farby alebo tri farebné zložky) potom vypíše príkaz `píš`.

Ako tento dialóg využijeme v našom programe? Do skupiny farebných tlačidiel pridáme ďalšie – môže vyzeráť približne takto:  Nezabudnite nastaviť, že patrí do skupiny k ostatným. Teraz už stačí do kolónky *Pri zapnutí* napísať `pisatko'nechfp pomôckaFarba pisatko'fp` (v praxi do slovenčiny: Zavolaj okienko na výber farby, na začiatku mu nastav farbu pera písatka a výsledok nastav ako novú farbu písatka.) A môžete sa vytešovať z farebných obrázkov.

Lenže – zas to má drobnú slabinu. Keď je stlačené červené tlačidlo, je jasné, že budeme kresliť na červeno. Ale ak je stlačená dúhová ceruzka, nastavená farba môže byť jemne ružová, tmavohnedá, nebičkovo modrá, či brontofialovohráškovaná, nemáme to ako rozoznať. Chcelo by to zabezpečiť, aby bolo vidieť, aká farba je práve nastavená aj nejak inak, než tak, že je stlačené patričné tlačidlo.


Úplne najlepšie by bolo, keby nastavenie bolo vidieť na našej všestrannej korytnačke písatko. Jej tvar zatiaľ obsahuje jediný (a aj to neviditeľný) pixel. To zmeníme. Už sme sa stretli s príkazom `nechTvar`, s ktorého pomocou môžeme meniť tvar korytnačky. Ako parameter sme mu zatiaľ dávali len meno súboru alebo obrázok, ktorým sa momentálny tvar korytnačky má nahradiť. Ako parameter mu ale môžeme zadať aj návod, ako sa má nový tvar korytnačky nakresliť, takže ak zadáte príkaz

`pisatko'nechTvar [opakuj 4 [dopredu 50 vpravo 90]]`

korytnačka nadobudne tvar úhľadného štvorčeka nakresleného jej aktuálnym perom. Pre naše účely bude úplne stačiť, ak nový tvar bude kreslený jednoduchým príkazom `bodka`, ktorý aktuálnym perom spraví jedinú bodku. Hrúbku pera radšej nastavte aspoň na 5, aby tam bolo aspoň niečo vidieť.

Takže čo treba urobiť? Treba prejsť všetky farebné tlačidlá a každému do kolónky *priZapnutí* pridať na koniec `pisatko'nechTvar [bodka]` pričom treba dať pozor, aby tam zostalo všetko to, čo tam bolo predtým.

Úloha č.1: Urobte to až potiaľto a vyskúšajte to.

Podíme sa teraz trochu pohrať s hrúbkou pera. Momentálne tam máme dve tlačidlá, s ktorých pomocou môžeme hrúbku meniť. Máme na výber medzi 1 a 5 pixelmi. Nie je to žiadna veľká sláva. Bolo by lepšie, keby sme si nejakým jednoduchým spôsobom mohli zvoliť ľubovoľnú hrúbku napríklad od 1 do 100. Na to sa dobre hodí taká vec, ako posúvatka.¹⁸ Takže obe hrúbkové tlačidlá zrušte a pridajte posúvatku. Stlačte ikonu , vyberte možnosť *Zvislý posúvač* a nejak

¹⁸ Spisovne posúvač, po anglicky slider.

vhodne ho umiestnite na plochu.

Teraz pôjde o to, posúvatko nejak šikovne nastaviť. Takže na neho kliknite pravým tlačidlom a ideme ho meniť. Najprv trochu zmeníme vzhľad (nečakane sa to robí v roletke *Vzhľad*). Výšku môžete jemne zväčšiť (asi na 180) a zaškrtnite *Priesvitné pozadie*, takže posúvatko bude vyzeráť približne tak, ako na obrázku vľavo. (Môžete stlačiť OK a pokochať sa.) Pokračujte v zmenách a presuňte sa do roletky *Základné*. Tam sa okrem mena (môžete nechať p1) nastavuje, aká je minimálna a maximálna hodnota posúvatka. Ak chceme, aby sa hrúbka pera menila od 1 do 100, minimálnu hodnotu nastavíme na 1 a maximálnu na 100. Posúvatko má svoju vlastnú hodnotu, ku ktorej sa môžete dopracovať príkazom *hodnota* a ktorá sa bude meniť podľa toho, ako posúvatko nastavíte. Úplne hore to bude 1 a úplne dole 100. (Keby ste potrebovali iný rozsah, stačí minimálnu a maximálnu hodnotu nastaviť inak.)

Teraz ešte treba posúvatku povedať, čo má spraviť, ak sa jeho hodnota zmení. Jednak treba písatku nastaviť novú hrúbku pera a jednak mu treba povedať, aby podľa novej hrúbky pera zmenilo svoj tvar. V základných nastaveniach mu do kolónky *priZmene* treba napísať

```
písatko'nechHp hodnota písatko'nechTvar [ bodka ]
```

A malo by to fungovať.

Úloha č.2: Vyskúšajte.

Na papieri môžeme pekne vidieť, ako sa veľkosť písatka mení. Ale niekedy je dôležité vedieť aj číselnú hodnotu hrúbky pera. Toto si môžeme zabezpečiť celkom jednoducho. Stačí pridať textové pole. Na to slúži tá istá ikona ako na pridávanie posúvatiek, akurát treba vybrať možnosť *Textový riadok*. Potom treba na ploche vyznačiť myšou obdĺžnik, kde chcete textové pole mať. Objaví sa vám políčko na text, ktoré bude vyzeráť približne takto: . Zmeňte jeho meno na *hrúbka* (áno, hádate správne, klik pravým tlačidlom...). Textové pole má podobne ako posúvatko hodnotu. (Jeho hodnotou je to, čo je v ňom napísané.) A táto hodnota sa dá meniť aj zvonka príkazom *nechHodnota*. Takže znovu zmeňte posúvatko a do kolónky *priZmene* na koniec dopíšte

```
hrúbka'nechHodnota hodnota
```

Spôsobí to ten príjemný efekt, že hodnota posúvatka¹⁹ sa zapíše ako hodnota textového poľa *hrúbka*. Môžete skúsiť.

No fajn. Máme posúvatko a keď ho posúvame, mení sa nám hodnota textového poľa. Lenže aj do textového poľa môžeme písať. Ale momentálne by ste hodnotu v textovom poli prepisovali márne. Na korytnačku *písatko* ani na posúvatko to nemá žiadny vplyv. Bolo by celkom príjemné, aby, keď do textového poľa zapíšeme nejakú rozumnú hodnotu, sa zmena prejavila na posúvatku aj na hrúbke pera. Nič ľahšie.

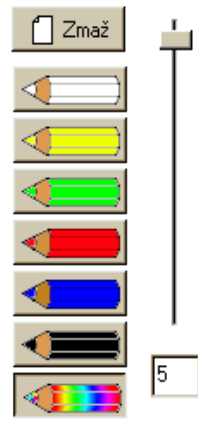
Zmeňte textové pole a do kolónky *priZmene* mu napíšte

```
ak ( zároveň ( číslo? hodnota ) ( hodnota <= 100 ) ( hodnota > 0 ) ) [p1'nechHodnota hodnota]
```

Ak zmeníte hodnotu textového poľa, overí sa, či je to číslo a nie nejaký abecedný blábol a ak to číslo je, či je v rozumnom rozsahu. Ak sú všetky podmienky splnené (požaduje to príkaz *zároveň*), hodnota posúvatka sa nastaví na hodnotu textového poľa. Celá ďalšia iniciatíva je na posúvatku. Vyvolá sa jeho udalosť *priZmene* (pretože jeho hodnota sa zmenila) a tá nastaví novú hrúbku pera a zmení korytnačku.

Keď váš program dotiahnete až sem, malo by to vyzeráť približne takto:










¹⁹ Ak chcem vedieť hodnotu objektu, s ktorým práve pracujem, stačí použiť príkaz *hodnota*. Ak potrebujem hodnotu iného objektu, treba použiť jeho meno a apostrof (napríklad *hrúbka'hodnota*).



Osemnásť kapitola

Korytnačka a ďalšie rafinované funkcie

Kreslítko je náš prvý väčší projekt a vydržal nám už na štvrtú kapitolu. A to máme pred sebou ešte všetky základné funkcie. Ale nebojte sa. Počas tejto kapitoly sa pokúsime doraziť ho.

Podme teda rýchlo na vec. Urobte si tlačidlá s menami **pero**,       
guma, vyplň, obdĺžnik, elipsa, násoska a písmo. Môžu vyzeráť približne tak, ako na obrázku vpravo, všetky budú prepínače a budú patriť do tej istej skupiny (takže zapnuté bude môcť byť najviac jedno z nich). Okrem nich si urobte ešte dve obyčajné tlačidlá s menami **uložiť** a **otvoriť**, s pomocou ktorých sa budú obrázky ukladať na disk a čítať z disku. Môžu vyzeráť napríklad takto:  .

Úloha č.1: Urobte.

Najprv si povieme o tom, ako sa dá zistiť, či je niektoré tlačidlo práve stlačené. Jednoducho – povie nám to funkcia **zapnutie**. Takže ak zadáme do príkazového riadku príkaz **píš pero'zapnutie**, v prípade, že je tlačidlo **pero** práve stlačené, sa vypíše **áno**, inak sa vypíše **nie**.

(Skúste si to pri oboch stavoch tlačidla.)

Doteraz naša korytnačka **písatko** mala celkom jednoduchú situáciu. Keď sa tlačidlo na myši stlačilo, začala kresliť (zavolať sa príkaz **pd**). Keď sa tlačidlo pustilo, kresliť prestala (zavolať sa funkcia **ph**).²⁰ Teraz to bude trochu zložitejšie. Správanie korytnačky sa bude meniť v závislosti od toho, ktorý nástroj je práve zvolený. Preto si otvorte okno korytnačky **písatko**, prepnite sa do roletky *Procedúry* a pridajte dve nové procedúry s menami **ĽavéDolu** a **ĽavéHore**. V procedúrach zatiaľ nebude nič, veci tam budeme pridávať iba postupne. Teraz sa prepnite do roletky *Udalosti*. Máte tam dve udalosti – **PriĽavomDolu** a **PriĽavomHore**. Zmeňte nastavenie tak, aby sa pri stlačení ľavého tlačidla zavolať procedúra **ĽavéDolu** a pri jeho uvoľnení procedúra **ĽavéHore**. Teraz už bude treba len dať procedúram **ĽavéDolu** a **ĽavéHore** taký obsah, aby všetky nástroje fungovali tak, ako majú.

V prípade, že je stlačené tlačidlo **pero**, je situácia rovnaká ako predtým. Do procedúry **ĽavéDolu** je teda treba pridať riadok

ak pero'zapnutie [pd]

a do procedúry **ĽavéHore** riadok

ak pero'zapnutie [ph]

Ešte drobná kozmetická úprava samotného tlačidla **pero**. Zavrite okno písatka, otvorte okno pera (pravý klik na tlačidlo...) a do kolónky *priZapnutí* zadajte príkaz **písatko'nechTvar [bodka]**. To pri každom zapnutí pera nanovo nastaví tvar písatka tak, aby vyzeralo rovnako, ako stopa, ktorú bude zanechávať.

Keď si to teraz vyskúšate, malo by to robiť presne to isté, čo pred chvíľou. A teraz môžeme pridávať nové funkcie. Ak chceme spojzduť gumu, musíme **pero** korytnačky prepnúť do režimu **guma**. (Pero sa môže nachádzať v jednom zo štyroch režimov: **PeroHore**, **PeroDolu**, **Guma** a **InverznéPero**.) Takže do procedúry **ĽavéDolu** pridajte riadok **ak guma'zapnutie [guma]** a do procedúry **ĽavéHore** riadok **ak guma'zapnutie [ph]**. Potom ešte zabezpečte, aby sa zmenil tvar písatka úplne rovnako, ako v prípade pera a bude fungovať už aj gumovanie.

S tlačidlom **vyplň** tiež nebudú väčšie problémy. Imagine totiž pozná priamo príkaz

²⁰ Detaily si pozrite v 14. kapitole

vypĺň, ktorý začne od miesta, kde sa korytnačka nachádza a všetky body rovnakej farby, ku ktorým sa dostane, vyplní farbou, ktorú má nastavenú ako FarbaVýplne. Jediný drobný zádrhel je v tom, že farba výplne môže byť odlišná od farby pera, takže ju pred samotným vyplňaním nastavíme tak, aby bola rovnaká. Do procedúry ĽavéDolu teda pridáme riadok ak vyplň'zapnutie [nechFv fp vyplň]. Pri zapnutí tlačidla vyplň ešte zmeníme tvar písatka (napr. písatko'nechTvar [nechHp 2 do 5 vz 10 do 5 vp 90 do 5 vz 10]) a môžeme vyplňať.

Pri kreslení obdĺžnikov a elíps siahneme po kresliacich funkciach iného druhu. Všetko kreslenie nám doteraz obstarávala korytnačka. Ale aj jednotlivé papiere majú svoje kresliace funkcie. Skúste napríklad zadať príkaz papier1'ppObdĺžnik [10 10 50 80]. Nakreslí sa vám obdĺžnik, ktorého jeden roh bude mať súradnice [10 10] a protiľahlý bude mať súradnice [50 80]. Papier má dokonca funkciu, s pomocou ktorej si môžeme natiahnuť obdĺžnik priamo myšou a ona nám povie jeho súradnice. Vyskúšajte zadať príkaz piš papier1'ppPružnýObdĺžnik a potom vyznačiť na papieri myšou obdĺžnik. Keď skončíte, obdĺžnik zmizne a Imagine vám vypíše súradnice dvoch jeho protiľahlých vrcholov. No a tieto dve funkcie sa dajú pre naše potreby dobre spojiť. A mimochodom, tieto „papierové“ funkcie majú znova samostatné nastavovanie farieb. Takže nový príspevok do procedúry ĽavéDolu bude vyzeráť asi takto:

```
ak obdĺžnik'zapnutie [ papier1'nechppFarba fp
                    papier1'ppObdĺžnik papier1'ppPružnýObdĺžnik ]
```

Situácia pri elipsách je veľmi podobná. Rovno pridajte aj toto:

```
ak elipsa'zapnutie [ papier1'nechppFarba fp
                   papier1'ppElipsa papier1'ppPružnáElipsa ]
```

Tvar korytnačky by mal byť čo najmenší, aby nezavadzala. Takže môžete pri oboch tlačidlách dať do kolónky *PriZapnutí* niečo ako písatko'nechTvar [nechHp 1 bodka] .

S tlačidlom násoska tiež nebude veľký problém. Stačí nastaviť farbu pera na farbu bodu na ktorom sa práve korytnačka písatko nachádza a zmeniť tvar písatka, aby sa prejavila zmena farby. Takže zase prídavok do ĽavéDolu, tentokrát takýto:

```
ak násoska'zapnutie [ nechFp farbaBodu
                    nechTvar [ nechHp 2 do 5 vz 10 do 5 vp 90 do 5 vz 10 ] ]
```

Rovnaký tvar je treba písatku nastaviť aj pri stlačení tlačidla násoska (to už je na vás).²¹ A je na svete ďalšia možnosť, ako si zvoliť farbu. Skrátka si zjédete niekam do svojho obrázku a farbu si „nasosnete“.

Podme sa teraz pozrieť na vkladanie textu do obrázku. Táto úloha sa dá riešiť mnohými spôsobmi. My ju skúsime riešiť takto: Pridáme pod tlačidlá ďalší textový riadok, rovnako ako sme pridávali textový riadok pre hrúbku pera (v kapitole 16). Nazvite si ho textík. Nie je potrebné, aby bol viditeľný stále, takže v tlačidle písmo nastavíme *priZapnutí* na textík'ukážMa a *priVypnutí* na textík'skryMa . Do kolónky *priZapnutí* ešte vpravíme príkaz na vynulovanie textu textík'nechHodnota " " a príkaz na zmenu tvaru písatka písatko'nechTvar [nechHp 1 bodka] . Teraz keď klikneme na tlačidlo písmo, objaví sa prázdne okienko na text a keď prepne nástroj na nejaký iný, okienko zase zmizne. Už len prinútiť korytnačku písatko, aby v prípade kliknutia do plochy daný text vypísala. To tiež nie je problém, na všetko sú funkcie. Konkrétne na výpis textu na plochu slúži korytnačke príkaz text. (Aké prekvapivé.) Takže do našej milej funkcie ĽavéDolu pripíšeme

```
ak písmo'zapnutie [ text textík'hodnota ]
```

21 Prečo sme vo funkcii ĽavéDolu nemuseli písať písatko'nechTvar ale stačilo napísať nechTvar ? Prečo keď meníme tvar písatka z niektorého tlačidla musíme písať písatko'nechTvar ?

a keď teraz klikneme na plochu (samozrejme pri zapnutom tlačidle písmo), korytnačka nám napíše text z riadku textík.

Úloha č.2: Dorobte funkčnosť všetkým vašim tlačidlám.

No a blížime sa do finále. Náš nástroj všeobecného umeleckého vyžitia je už takmer dokonalý. Celé to má ale jednu slabinu. Akonáhle chceme kresliť nový obrázok, predošlý je nenávratne stratený. Chýba tomu systém ukladania na disk. Takže ho podíme dorobiť.

Venujme sa najprv tlačidlu uložiť. Imagine obsahuje skvelú funkciu uložiťObrázok. Tejto funkcii treba zadať dva parametre – prvý je meno súboru, do ktorého sa má obrázok uložiť. Druhý parameter je samotný obrázok. Funkcia uložiťObrázok je natoľko dokonalá, že podľa mena súboru vyberie správny formát ukladania. Súbor s koncovkou .bmp uloží ako bitmapu, súbor, ktorý má koncovku .jpg uloží ako komprimované jpg-čko. Keď teda zadáte príkaz

```
uložiťObrázok "dielo.bmp papier1'pozadie
```

to, čo ste zatiaľ na papier1 namachlili sa očitne na harddisku v súbore *dielo.bmp* (pravdepodobne v adresári *Program Files\Imagine\Obrázky*) a môžete si to pozrieť napríklad s pomocou programu Gimp, či iného programu schopného pracovať s obrázkami.

Obrázok by sme uložiť vedeli. Aby to vyzeralo úplne profesionálne, chcelo by to nejaké menu na výber mena súboru. Ako sa dá čakať, Imagine má funkciu aj na toto. Skúste zadať príkaz

```
píš pomôckaUložGrafiku []
```

Objaví sa vám výberové menu. Skúste vybrať nejaký súbor alebo zadať nejaké meno. Funkcia vráti meno tohto súboru aj s celou prístupovou cestou.

No a tieto dve funkcie pekne spojíme dohromady. Tlačidlu uložiť povieme, aby *priZapnutí* zavolalo procedúru uložiť. Prepne sa do roletky *Procedúry* a pridáme procedúru uložiť. A do tejto procedúry uložíme nasledujúci kód:

```
urobTu "súbor pomôckaUložGrafiku []  
ak prázdne? :súbor [ukonči]  
uložiťObrázok :súbor papier1'pozadie
```

Najprv zavoláme menu na výber mena súboru a vybrané meno uložíme do premennej *súbor*. Ak je premenná prázdna (dialógové okno sme zavreli bez toho, že by sme nejaké meno zvolili), nič sa neudeje. Ak sa nám meno zvoliť podarilo, obrázok sa uloží pod zvoleným menom.

Podobné to bude aj s tlačidlom otvoriť. Nastavte *PriZapnutí* na volanie procedúry otvoriť, ktorú pridáte ako procedúru tohto tlačidla. Využijeme dialóg na výber obrázka na čítanie, ktorý zavoláme príkazom *pomôckaČítajGrafiku []*. Samotný obrázok načítame príkazom *obrázokZoSúboru* a nastavíme ho ako nové pozadie pre *papier1*. V tomto ale môže byť problém – rozmer papiera sa pri zmene pozadia prispôsobí nastavovanému pozadiu. Preto si starý rozmer uložíme do pomocnej premennej a po načítaní obrázka ho opäť nastavíme. Takže procedúra otvoriť bude vyzeráť takto:

```
urobTu "súbor pomôckaČítajGrafiku []  
urobTu "rozmer papier1'veľkosť  
ak prázdne? :súbor [ukonči]  
papier1'nechPozadie obrázokZoSúboru :subor  
papier1'nechVeľkosť :rozmer
```

No a to je všetko. Kresliaci program je hotový. Nie, že by nemal chyby (skúste napríklad stlačiť farebnú ceruzku a potom žiadnu farbu nevybrať a okno zavrieť) a že by sa na ňom nedalo už nič vylepšiť (napríklad možnosť meniť pozadie, pridať prácu s obrázkami rôznych veľkostí, dorobiť lupu a podobne) ale už teraz sa s tým dajú kresliť celkom pekné veci. A ďalšie vylepšovanie je na vás.

Úloha č.3: Dorobte a ak viete, tak vylepšite.



Dóm sv. Martina

Zmaž

White pencil icon

Yellow pencil icon

Green pencil icon

Red pencil icon

Blue pencil icon

Black pencil icon

Rainbow pencil icon

Vertical slider with a grey knob at the top and a box containing the number 12 at the bottom.

Eraser icon | Pencil icon | Fill bucket icon | Red square icon | Red circle icon | Lasso icon | Text 'A' icon | Save icon | Undo icon

Dóm sv. Martina

Devätnásta kapitola

Korytnačka a hodiny

V tejto kapitole sa budeme zaoberať miniprojektíkom – hodinami. Vyrobíme si klasické ručičkové hodinky, ktoré nám názorným spôsobom budú ukazovať systémový čas. Projekt si môžeme uložiť na plochu a s jeho pomocou mať prehľad, kedy už konečne zazvoní alebo ako dlho už za tou mašinou sedíme, prípadne sa ním môžeme pochváliť vzdialenejšiemu príbuzenstvu, nech vie, aký poklad to má v rodine.

Začneme procedúrou **Štart**. Jednak, že je to pekné meno pre procedúru, ktorou sa ide začínať, druhak, že ak sa v Imaginovskom projekte nachádza procedúra s týmto ľubozvučným menom, tak sa vždy pri otvorení projektu automaticky spustí, takže si nemusíme drať prsty pri vypisovaní príkazov do príkazového riadku. Keď sa procedúra píše, máme dve možnosti. Môžeme používať príkazy, ktoré Imagine pozná a nejak to z nich poskladať. (Tak sme to v podstate robili doteraz.) Je tu ale druhá možnosť – môžeme používať príkazy, o ktorých nemá Imagine ani šajn. Problém je v tom, že si ich potom musíme naprogramovať sami. Napriek tomu ale tento prístup má niečo do seba, takže sa ho budeme držať. Procedúra **Štart** teda bude vyzerať takto:

```
viem Štart
všetkoPomaž
nakresliHodinky
vyrobKorytnačky
zapniHodiny
koniec
```

Aspoň približne je jasné, čo jednotlivé procedúry majú robiť. Prvá zmaže, čo môže. Druhá nakreslí ciferník. Tretia vyrobí tri ručičky – korytnačky a štvrtá to celé spustí. Každú z týchto procedúr môžeme písať do istej miery nezávisle na ostatných a keď sa nám neskôr niektorá z nich znepáči, môžeme ju prerobiť bez toho, že by sme poškodili ostatné.

Takže od začiatku. Procedúra **všetkoPomaž** bude celkom jednoduchá. Zrušíme všetky korytnačky a zmažeme plochu.

```
viem všetkoPomaž
zrušObjekt všetky
zmaž
koniec
```

Procedúra **nakresliHodinky** by tiež nemala robiť vážnejšie problémy. Stačí vhodnou farbou nakresliť dokola dvanásť čiarok. Predtým si ale nastavte stránku tak, aby jej rozmery boli 320 × 320 bodov a počiatok súradnicovej sústavy bol v bode [160, 160]. (*Okno Pamäť*, *dvojklik na Stránka 1, záložka Vzhľad a tam Šírka, Výška a súradnice počiatku*) Samotné kreslenie môže vyzerať napríklad takto:

```
viem nakresliHodinky
nová "korytnačka [ meno Klementína ]
odteraz "Klementína
skry
nechFp "bledoSivá nechHp 10
opakuj 12 [ ph do 120 pd do 20 ph vz 140 vp 30]
zrušObjekt "Klementína
koniec
```

Vzhľadom k tomu, že sme si príkazom **všetkoPomaž** zmazali všetky korytnačky, na to, aby sme mohli vôbec niečo nakresliť si musíme nejakú zas vytvoriť. Nazveme ju **Klementína**, spravíme ju aktívnou, skryjeme ju, nech ju nie je vidieť, nastavíme farbu a hrúbku pera, nakreslíme 12 čiarok (pochopte, ako to kreslenie vlastne funguje!!!) a korytnačku **Klementínu** zase zrušíme.

Teraz príde na vyrábanie korytnáčiek – ručičiek. Stačí im nastaviť meno a tvar. Tvar opíšeme s pomocou príkazov **Imaginu** – s tým sme sa už stretli v našom veľkom výtvarnom projekte.

```
viem vyrobKorytnačky
  nová "korytnačka [
    meno Malá
    tvar [ nechHp 10 nechFp 0 ph vz 10 pd do 100 ] ]
  nová "korytnačka [
    meno Veľká
    tvar [ nechHp 5 nechFp 0 ph vz 20 pd do 150 ] ]
  nová "korytnačka [
    meno Sekundová
    tvar [ nechHp 2 nechFp "červená ph vz 20 pd do 150 ] ]
koniec
```

Pri kreslení každej ručičky sme sa posunuli o kúsok dozadu, nech ručičky kúsok prečnievajú za miesto, okolo ktorého sa otáčajú. Sekundovú ručičku sme nakreslili červenou.

Ostáva nám už len posledná procedúra, ktorá spustí proces otáčania ručičiek. Na to, aby sme zistili stav systémových hodín slúži príkaz **čas**. Ak zadáte príkaz **píš čas**, Imagine na vás vychrlí zoznam štyroch čísel, napríklad **21 8 6 740**, kde jednotlivé čísla znamenajú hodiny, minúty, sekundy a tisíce sekundy. Ak by sme potrebovali zistiť počet minút, spravili by sme to príkazom **prvok 2 čas** (detaily pozrite v trinástej kapitole).

Sekundovú ručičku bude najjednoduchšie nastaviť. Za 60 sekúnd sa musí otočiť o 360 stupňov. To je 6 stupňov za sekundu. Takže do správnej polohy ju nastaví príkaz

```
sekundová'nechSmer 6 * (prvok 3 čas)
```

Kde (prvok 3 čas) je aktuálny počet sekúnd. Minútová ručička to má trochu komplikovanejšie. Ak nechceme, aby sa pohybovala iba raz za minútu, musíme vypočítať, aká časť minúty uplynula. Podobne (a ešte komplikovanejšie) to je s hodinovou ručičkou.

Okrem správneho výpočtu smeru ručičiek sa musíme postarať o to, aby sa správny smer nastavoval v pravidelných intervaloch – musíme spustiť proces, ktorý sa o to bude starať. Oba tieto problémy rieši funkcia **zapniHodiny**:

```
viem zapniHodiny
  každých 100 [
    sekundová'nechSmer 6 * (prvok 3 cas)
    veľká'nechSmer ((prvok 2 cas) + (prvok 3 cas / 60)) * 6
    malá'nechSmer ((prvok 1 cas) + (prvok 2 cas / 60) + (prvok 3 cas / 3600)) * 30
  ]
koniec
```

Príkaz **každých 100** spôsobí, že proces uvedený v hranatých zátvorkách sa vykoná každých 100 milisekúnd (teda desaťkrát za sekundu). Ručičky sa do správnej polohy budú nastavovať takmer nepretržite a hodinky pôjdu.

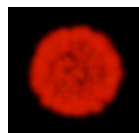
Aby bol projekt úplne dokonalý, možno by bolo vhodné do procedúry **Štart** pridať na začiatok ešte funkciu

Dvadsaata kapitola

Korytnačka a potomkovia

Konečne by to chcelo spraviť nejakú hru. Niečo jednoduchšie, s čím sa dalo hrať, sme už urobili aj v predošlých kapitolách, ale chcelo by to spraviť niečo aspoň trochu zložitejšie. Na začiatok skúsime urobiť variáciu na klasiku – raketka ostreľuje asteroidy. Vždy, keď niektorý asteroid zasiahne, ten sa rozpadne na štyri menšie, každý z nich sa po zásahu rozpadne na štyri ešte menšie a až tie ďalší zásah definitívne zlikviduje. Predbežne nám pôjde iba o to, aby to strieľanie fungovalo a aby sa meteority rozpadali. Keď bude tento polotovar hotový, ďalšie vlastnosti (napríklad nutnosť uhýbať pred meteoritmi alebo ich zasiahnuť skôr, než zasiahnu mňa, stopovanie času potrebného na zlikvidovanie všetkých meteoritov, výpis správy o šťastnom ukončení hry, tabuľka najlepších dosiahnutých výsledkov, ...) sa dajú dorobiť aj dodatočne.

Dôležitou fázou vývoja hry je vytváranie animácií jednotlivých postáv, ktoré sa v hre pohybujú (v programátorskom žargóne sa takáto postavica hry volá avatar). Budeme potrebovať nejaké asteroidy (imaginovská korytnačka *Asteroidy.lgf* niekoľko pekných obsahuje), raketu, strelu a výbuch. Buď si ich urobte sami, alebo použite tie, ktoré sme vyrobili my. Náš výbuch obsahuje 9 fáz (s týmto faktom budeme počítať, takže ak si robíte vlastné animácie, tiež si ich urobte toľko, alebo zmeňte patričné miesta v kóde.)



Okrem toho by to chcelo ešte nejaký rachot, keď sa nejaký meteorit zasiahne. (Ľudia znalejší veci vedia, že keďže vo vesmíre nie je vzduch, nie je tam ani rachot, ale hráči ocenia.) Buď si nejaký nájdite na internete, alebo použite **náš**. Korytnačky umiestnite do podadresára Imaginu s názvom *Obrazky* a rachot do toho istého adresára, v ktorom máte projekt.

Keď máme všetky súbory na správnych miestach, môžeme sa začať venovať programovaniu. A práve teraz nastal čas spomenúť, prečo má táto kapitola taký zvláštny názov. Veď na prvý pohľad korytnačie potomstvo s vesmírnou strieľačkou nič spoločné nemá.

S tým potomstvom je to totiž tak – korytnačka je potvora, ktorá má rôzne vlastnosti (napríklad tvar, smer alebo farbu pera). Lenže občas aj nejaké vlastnosti nemá. Napríklad by sme chceli, aby asteroid mal určitú rýchlosť. Môžeme použiť normálnu korytnačku a zriadiť jej premennú (takto sme to robili doteraz), lenže keď nám tam tých asteroidov bude poletovať pätnásť, bude treba takú premennú zriadiť každému z nich, čo je dosť pracné. Ale to môžeme vyriešiť inak. Môžeme si urobiť novú triedu – asteroid, ktorý bude potomkom triedy korytnačka. To, že je jej potomkom znamená, že podedí jej vlastnosti. A okrem nich môže mať aj nejaké vlastné (teda napríklad tú rýchlosť), prípadne niektoré rodičovské vlastnosti môže predefinovať podľa seba.

Podme teda vytvoriť novú triedu. Použijeme na to (prekvapivo) nasledujúci príkaz:

```
nováTrieda "korytnačka "asteroid []
```

nastavOkno, ktorá Imagine nastaví tak, aby nezaberal na ploche veľa miesta a neukazoval zbytočnosti. Môžete si ho dorobiť napríklad takto:

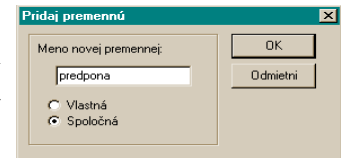
```
viem nastavOkno
  skryHlavnýPanel
  bezPlochyVýpisov
  zavriPamäť
  hlavnéOkno'nechVeľkosť [330 370]
koniec
```

Čo jednotlivé funkcie robia, je viacmenej zrejmé z ich názvov. Ak chcete vedieť podrobnosti, pozrite si Pomocníka.

Po príkaze `nováTrieda` nasleduje meno rodičovskej triedy (v tomto prípade trieda `korytnačka`) a potom meno novej triedy. Na konci sú hranaté zátvorky v ktorých je možné rovno určiť nové vlastnosti. To ale tentokrát spravíme v okne *pamäť*, takže budú prázdne.

Po vykonaní uvedeného príkazu sa nám v okne *pamäť* objaví taká vyblednutá korytnačka – to je ONA, naša nová trieda s názvom `asteroid`. Smelo na ňu dvojkliknite. Ukáže sa vám podobné okno ako pri korytnačkách, toto však obsahuje iba tri záložky: *Udalosti*, *Premenné* a *Procedúry*. Vyberte *Premenné* a hor sa pridávať.

Prvá premenná, ktorú pridáme, sa bude od ostatných v drobnom detaile líšiť. Bude to totiž premenná, ktorá bude spoločná celej triede. (Teda to nebude tak, že by si ju každý asteroid mohol nastaviť po svojom.) Ako vidíte na obrázku vľavo, treba jej nastaviť možnosť *Spoločná*. Bude to premenná `predpona`, ktorá určuje, ako sa budú jednotlivým objektom danej triedy automaticky generovať mená. Ak jej hodnotu nastavíte na `ast`, tak sa novým objektom triedy `asteroid` budú dávať mená `ast1`, `ast2`, ... (ak explicitne neurčíte iné meno). Asi ste už uhádli, že trieda `korytnačka` má premennú `predpona` nastavenú na `k`.



Udalosti	Premenné	Procedúry
[S] predpona	ast	
[V] poz	?	
[V] smer	?	
[V] záber	6	
[V] tvar	asteroidy	
[V] reagujeNaZrážku	áno	
[V] rýchlosť	1	

Ostatné premenné budú vlastné. Budú to buď nové vlastnosti (ako premenná `rýchlosť`), alebo vlastnosti korytnačky, ktorým chceme pri vzniku nastaviť novú štandardnú hodnotu. Vytvorte a nastavte ich rovnako, ako vidíte na obrázku.

Všetky veci v tabuľke až na dve výnimky sú čitateľné celkom jednoducho. Chceme, aby mal nový asteroid tvar `asteroidy` (konkrétne záber 6), chceme, aby reagoval na zrážku a aby mal rýchlosť 1. Otázne sú akurát tie otázniky. Otáznik v Imagine slúži ako univerzálny generátor náhody. Príkaz `nechSmer ?` znamená niečo podobné, ako príkaz `nechSmer náhodne 360` a `nechPoz ?` nastaví korytnačku na náhodné miesto na ploche. Takže tými otáznikmi Imaginu povieme, že nový asteroid má mať náhodnú polohu a smer.

Úloha č.1: Urobte triedu `asteroid` a potom niekoľkokrát zadajte príkaz `nový "asteroid []` Čo to bude robiť?

Úloha č.2: Rovnako, ako sme spravili triedu `asteroid`, teraz vyrobte triedu `strela`. Premennú `predpona` nastavte na `strela`, premennú `tvar` nastavte na `strela`, premennú `reagujeNaZrážku` nastavte na `áno` a pridajte premenné `rýchlosť` (hodnota 6) a `vek` (hodnota 0).²² Dajte pozor na to, aby premenná `predpona` bola spoločná.

Úloha č.3: Urobte triedu `výbuch`. Premennú `predpona` nastavte na `bum`, tvar na `výbuch`, premennú `autoAnimovanie` na `nie`²³, fáza na 1 a `reagujeNaZrážku` na `nie`.

A keď vám to vytváranie nových tried tak ide, tak...

Úloha č.4: Urobte triedu `raketa`, s predponou `raketa` a tvarom `raketa`, ktorá nebude reagovať na zrážku.

Tieto triedy nám už našťastie budú stačiť.

Na `asteroidy`, `strely` a `výbuchy` budeme mať tri zoznamy, každý v jednej globálnej premennej. Ich mená budú `(prekvapivo)` `asteroidy`, `strely` a `výbuchy`. Predbežne budú prázdne. Vytvorte si ich. (Príkazom `urob "asteroidy []` podobne aj ostatné.) No a chceli by sme dosiahnuť, aby sa každý asteroid pri svojom vzniku sám zapísal do zoznamu `asteroidov`.

Dá sa to spraviť s pomocou udalosti `priVytvorení`. To je udalosť, ktorá sa vyvolá vždy

²² Strela nám nemôže poletovať po hracej ploche donekonečna, takže budeme musieť sledovať jej vek a po nejakom čase ju zrušiť aj keď nič netrafí.

²³ Nechceme, aby sa výbuch ukazoval automaticky stále dokola, pretože by bol problém ho v správnom okamihu zrušiť. Radšej vypneme `autoAnimovanie` a budeme nastavovať jeho jednotlivé fázy sami.

práve raz – pri vytvorení daného objektu. Takže našej triede `asteroid` povieme, aby vtedy, keď nastane udalosť `priVytvorení` vykonal objekt príkaz

```
urob "asteroidy vložPosledný mojeMeno :asteroidy
```

Príkaz `mojeMeno` vráti meno objektu, ktorý ho zavolať (takže napríklad `k1.mojeMeno` bude `k1`). Príkaz `vložPosledný` vyrobí v tomto prípade zoznam, ktorý vznikne vložením mena objektu na koniec zoznamu `asteroidov`, takže to spraví presne to, čo potrebujeme.

Úloha č.5: Pridajte triede `asteroid` obsluhu udalosti `priVytvorení`. Vyskúšajte zadať príkaz `nový "asteroid []` Aká bude hodnota premennej `asteroidy` ?

Úloha č.6: Pridajte podobné spracovanie udalostí `priVytvorení` aj triedam `strela` a `výbuch`. Dbajte na to, aby sa zapísali do svojich vlastných zoznamov.

Všetko starostlivo uložte, pokračovanie v ďalšej kapitole :-)

Dvadsaata prvá kapitola

Rozpohybované korytnačky

Keď sme začali vyrábať našu hru, začali sme tým, že sme si vyrobili tvary „postavičiek“ (áno, aj výbuch je postavička) a zriadili sme si objekty, ktoré budú jednotlivé postavy reprezentovať – konkrétne to boli objekty raketa, asteroid, strela a výbuch. Vyrobili sme si aj globálne prístupné zoznamy asteroidy, strely a výbuchy a zariadili sme to tak, aby sa patričný objekt hneď pri vzniku zaradil do svojho zoznamu. Teraz bude treba zariadiť, aby sa to všetko hýbalo, zrážalo a vybuchovalo podľa našich predstáv.

Začnime s asteroidom. Vyrobíme mu úplne jednoduchú procedúru, ktorú nazveme pohni a bude vyzeráť takto:

```
viem pohni
  dopredu rýchlosť
koniec
```

Skrátka asteroid sa pohne svojou rýchlosťou o kúsok dopredu.


Len tak na skúšku si vyrobte desať asteroidov (príkaz opakuj 10 [nový "asteroid []]). Keď chceme pohnúť všetkými asteroidmi, spravíme to príkazom

```
preKaždú :asteroidy [ pohni ]
```

Tento príkaz je zo slovenčinárskeho hľadiska pomerne podivný. Asteroid je totiž rodu mužského. Lenže ako sme si povedali v predošlej lekcii, asteroid je potomkom korytnačky, tým pádom je v podstate tiež korytnačka. A čo sa korytnačiek týka, je ženský rod na mieste. No a príkaz **preKaždú** urobí pre každú korytnačku zo zoznamu, ktorý mu predhodíte to, čo je uvedené v hranatých zátvorkách.

No fajn. Asteroidy sa nám o kúsok pohli. Ako ale dosiahnuť, aby sa hýbali stále? Na to treba spustiť proces. A proces sa spustí celkom jednoducho. Stačí napísať

```
každých 20 [ preKaždú :asteroidy [ pohni ] ]
```

Vznikol proces. A tento proces si zobral na starosť, že každých 20 milisekúnd vykoná to, čo sme mu povedali v hranatých zátvorkách. A tak sa asteroidy hýbu, až kým nestlačíme .

Úloha č. 0,5: Vyskúšajte.

Teraz si zriadime ešte jeden zoznam, ktorý bude plniť funkciu smetiska. Do neho budeme zapisovať objekty zrelé na zničenie (a časom nejakú funkciu poveríme, aby ich zlikvidovala a upratala). Globálnu premennú, do ktorej ten zoznam uložíme, nazveme zničené. Keď si ju urobíte, môžeme vyrobiť procedúru pohni pre objekt strela.

Pre strelu musíme zabezpečiť, aby sa hýbala. Ale nemala by sa hýbať úplne do nekonečna. Treba zabezpečiť, aby časom vyhynula, aj keď nič netrafí. Presne za tým účelom sme triede strela minule zriadili premennú vek. Takže funkcia pohni pre strely bude vyzeráť takto:

```
viem pohni
  nechVek vek + 1
  ak vek > 55
  [
    urob "zničené vložPosledný mojeMeno :zničené
  ]
  dopredu rýchlosť
koniec
```

Najprv sme vek zvýšili o 1, v prípade, že vek dosiahol 55 (a teda korytnačka rýchlosťou 6 prešla

330 krokov) tak si povieme, že už bezcieľne preletela dosť a zaradíme ju do zoznamu na zničenie. No a nakoniec so strelou smelo pohneme o rýchlosť.

Kým nemáme funkciu na správu zoznamu zničené, žiadne vyhynutie rakety sa neuskutoční a bude sa do zoznamu pridávať stále znova (uložte si projekt, vyrobte si proces na pohybovanie strelami, nechajte ho chvíľu bežať a potom sa pozrite, čo máte v premennej zničené), ale nebojte sa, to sa časom zlepší.

Ďalšia dôležitá vec, ktorú musí strela urobiť, je, že si musí všimnúť, ak niečo trafila. Na to máme k dispozícii udalosť priZrážke. Aj asteroidy, aj strely sú citlivé na zrážku, takže ku kolízii medzi nimi môže dôjsť. Strele zriadime procedúru havária, ktorú zavesíme na udalosť priZrážke. Od procedúry havária budeme chcieť, aby do zoznamu zničených objektov zaradila samotnú strelu, potom všetky asteroidy, s ktorými sa práve prekrýva a nakoniec na mieste, kde sa strela práve nachádza vyrobila výbuch. Takže to môže vyzeráť napríklad takto:

```
viem havária
  urob "zničené vložPosledný mojeMeno :zničené
  urob "zničené veta :zničené prekrývajúMa
  ( novýN "výbuch "poz poz )
koniec
```

Použili sme nový príkaz prekrývajúMa, ktorý zistí všetky korytnačky, ktoré strelu prekrývajú a vráti ich pekne poukladané v zozname. Ten zoznam potom s pomocou príkazu veta kompletne prilepíme k zoznamu korytnáčiek na zničenie.

Úloha č. 0,75: Urobte to potiaľto.

Podíme sa teraz venovať výbuchom. Minule sme výbuchom nastavili hodnotu premennej autoAnimovanie na nie. Musíme sa preto postarať, aby sa jednotlivé fázy výbuchu vykreslili tak, ako majú a keď už výbuch dovybuchuje, aby sa zlikvidoval a viac na obrazovke nestrašil. Aj výbuchu spravíme procedúru pohni, ktorá bude vyzeráť takto:

```
viem pohni
  ak fáza = 8
  [
    urob "zničené vložPosledný mojeMeno :zničené
  ]
  nechFáza fáza + 1
koniec
```

Ak je výbuch v predposlednej (ôsmej) fáze, vyhodíme ho na smetisko. Potom zapneme ďalšiu fázu. Aj keď výbuch môže byť už na smetisku, likvidačná procedúra príde k slovu až o chvíľu, takže posledná fáza bude ešte nejaký čas zobrazená.

Úloha č. 0,9: Potiaľto.

No a nakoniec ostáva samotná raketa. Jej úlohou je vypúšťať strely. Vyrobíme jej procedúru streľ, ktorá vypustí strelu tým smerom, ktorým je otočená raketa. Procedúra bude potvora a ak už sú dve rakety vypustené, ďalšiu nevypustí:

```
viem streľ
  ak počet :strely < 2
  [
    ( novýN "strela "smer smer )
  ]
koniec
```

Raketu budeme ovládať s pomocou kláves. Jedna z možností, ako to urobiť je, že jej zriadime novú premennú – premennú ponukaKlávesov. Jej obsah bude takýto:

[vľavo [vľavo 3] vpravo [vpravo 3] medzera [strel]]

Od tohto momentu raketa začne reagovať na to, čo stlačíte. Ak stlačíte šípku vľavo, otočí sa o tri stupne doľava, ak stlačíte šípku vpravo, otočí sa o tri stupne doprava a ak stlačíte medzeru, zavolá sa procedúra strel.

Úloha č.1: Potiaľto.

Táto kapitola bola ľahká, bola v nej iba jedna úloha, takže si to všetko pekne uložte, pokračujeme nabudúce. :) Budeme robiť hlavnú procedúru, ktorá spôsobí, že sa to už bude dať hrať.

Dvadsaata druhá kapitola

Explodujúce korytnačky

V tejto kapitole sa pokúsime spraviť hlavnú procedúru, ktorá bude zodpovedná za to, aby sa asteroidy hýbali, výbuchy vybuchovali a strely triafali. Majte na pamäti, že vyrábame jednu veľkú procedúru. Jednotlivé jej časti budeme priebežne komentovať. Nechávame na dôvtipnosti čitateľa, aby odlíšil text, ktorý má do procedúry vložiť od komentárov.

Pripomeňme, že procedúra hlavná bude mať na starosti každým objektom pohnúť iba o kúsok. Neskôr spravíme proces, ktorý ju bude v pravidelných intervaloch spúšťať. Takže

```
viem hlavná
```

Najprv sa budeme zaoberať zoznamom zničené. Použijeme príkaz `prePrvky`, ktorý do premennej `:l` bude postupne dosadzovať všetky prvky zo zadaného zoznamu a s touto hodnotou vykoná všetky príkazy uvedené v hranatej zátvorke.

```
prePrvky "l :zničené [
```

Ak je prvých šesť písmen názvu objektu „strela“, tak sa jedná o strelu.

```
ak prvok [1 6] :l = "strela
[
```

Zničenú strelu vyradíme zo zoznamu striel a zrušíme.

```
urob "strely bezVýskytov :l :strely
zrušObjekt :l
]
```

Rovnako, ako s vyradenými strelami, teraz naložíme s výbuchmi, ktoré už dovybuchovali.

```
ak prvok [1 3] :l = "bum
[
urob "výbuchy bezVýskytov :l :výbuchy
zrušObjekt :l
]
```

Teraz nás čaká najväčšia zábava. Práve zlikvidovaný asteroid totiž nemusí len tak zmiznúť, ale pri zásahu sa môže rozpadnúť na menšie.

```
ak prvok [1 3] :l = "ast
[
urob "asteroidy bezVýskytov :l :asteroidy
pre :l [
```

Keď likvidujeme asteroid, musíme sa pozrieť, aký má záber. Najväčší asteroid má 6, menší 2 a najmenší 1. Ak sa práve nejedná o najmenší z nich, vyrobíme miesto neho štyri menšie.

```
ak záber <> 1
[
ak záber = 6
[ urobTu "nový 2]
ak záber = 2
[ urobTu "nový 1]
opakuj 4 [
( novýN "asteroid
"poz poz
```

```
"rýchlosť rýchlosť + 1
"smer náhodne 360
"záber :nový)
```

```
]
```

```
]
```

```
]
```

Zrušíme zlikvidovaný asteroid.

```
zrušObjekt :!
```

```
]
```

```
]
```

Keď sme po zničených objektoch upratali, môžeme zoznam vyprázdniť

```
urob "zničené []
```

Nakoniec pohneme všetkými korytnačkami, ktoré sú stále v zoznamoch.

```
preKaždú :strely [pohni]
```

```
preKaždú :asteroidy [pohni]
```

```
preKaždú :výbuchy [pohni]
```

koniec

Teraz ešte spravíme procedúru **Štart** (ak projekt obsahuje procedúru s týmto menom, tak sa táto procedúra automaticky spustí po otvorení projektu), ktorá spraví nejaké úvodné nastavenia a vytvorí toľko očakávaný proces:

```
viem štart
```

Nastavíme farbu pozadia na čiernu a zrušíme všetky objekty.

```
nechFarbaPozadia 0
```

```
zrušObjekt všetky
```

Vyrobíme si zoznamy, ktoré potrebujeme a zatiaľ ich nastavíme na prázdne.

```
urob "asteroidy []
```

```
urob "strely []
```

```
urob "výbuchy[]
```

```
urob "zničené []
```

Urobíme si novú raketu a asteroid.

```
nová "raketa []
```

```
nový "asteroid []
```

Vyrobíme proces, ktorý každých 20 milisekúnd zavolá procedúru **hlavná**. Proces nazveme **hlavná**.

```
(každých 20 [hlavná] "hlavná)
```

```
koniec
```


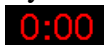
Napíšeme **štart** a môžeme sa hrať.

Dvadsaata tretia kapitola

Korytnačky a časomiera

Skôr, než sa začneme venovať hlavnej téme tejto kapitoly, spravíme najskôr jednu drobnú zmenu – konečne využijeme súbor `explos.wav`. Otvoríme triedu `strela` a na koniec procedúry `havária`, ktorá obhospodaruje zrážku strely s asteroidom, pridáme príkaz `hrajZvuk "explos.wav"`. Spôsobí to, že pri každom zásahu sa vesmírnym vzduchoprázdnom ozve riadny rachot.

Teraz sa môžeme začať venovať hlavnej téme tejto kapitoly a tou je časomiera a veci s ňou súvisiace. Totiž – je pekné, že môžeme vystrieľať všetky asteroidy. Aj pochváliť sa s tým môžeme. Ale oveľa lepšie, než „vystrieľal som všetky asteroidy“ znie „vystrieľal som všetky asteroidy za minútu a dvadsaťštyri sekúnd a uhral som nový rekord“. Takže teraz sa budeme venovať tomu, aby Imagine odmerala čas, ktorý spotrebujeme na vystrieľanie všetkých asteroidov, najlepšie časy nejakým spôsobom ukladal a pochválil nás, keď nejaký najlepší čas ešte vylepšíme.

Prvá vec, ktorú urobíme je, že do plochy vložíme text. Text sa vkladá tlačidlom . Vložte ho niekde vľavo hore, nastavte mu nejakú rozumnú veľkosť (aspoň 18), farbu písma zmeňte na červenú a vyberte nejaký pekný font (podľa možností nie Wingdings). Do textu preventívne nastavte hodnotu 0:00 a okraje rámika nastavte tak, aby sa do neho vošli aj trocha dlhšie časové údaje. Keď s tým všetkým budete hotoví, stlačte tlačidlo `Esc`, a potom na text kliknite pravým tlačidlom myši, aby ste mohli zmeniť niektoré ďalšie nastavenia. Najprv zmeňte samotné meno textu z `text1` na `hodiny`. Potom zmeňte farbu pozadia na čiernu, aby rámik splýval s pozadím a nebolo ho vidieť. Keď skončíte, malo by to vyzeráť približne takto:  a keď zadáte kontrolný príkaz

```
hodiny'nechHodnota "8:45
```

ukazovaný čas by sa mal zmeniť.

Čo sa samotného merania času týka, Imagine pri štarte spustí časomer, ktorý meria čas od štartu v tisícinach sekundy. Jeho hodnotu si môžete dať vypísať príkazom `piš časomer`. Hodnota časomeru sa dá kedykoľvek vynulovať príkazom `vynulujČasomer`. Čas sa potom meria od posledného vynulovania. Spravíme to teda tak, že pri štarte časomer vynulujeme a urobíme si procedúru, ktorá nám podľa neho bude vedieť správne nastaviť text hodiny. A potom si zriadime proces, ktorý túto procedúru bude v pravidelných intervaloch volať.

Procedúra na zobrazenie času sa bude nazývať `ukážČas`. Do procedúry štart teda pridáme riadky:

```
vynulujČasomer  
( každých 500 [ukážČas] "ukážČas )
```

A samotná procedúra `ukážČas` bude vyzeráť takto (komentáre do nej nepíšte):

```
viem ukážČas  
  urobTu "sekundy cPodiel časomer 1000  
        ;Zistili sme si, koľko sekúnd uplynulo od vynulovania časomera  
  urobTu "minúty cPodiel :sekundy 60  
        ;Počet minút je počet sekúnd deleno 60. cPodiel vypočíta celočíselný  
        ;podiel  
  urobTu "sekundy zvyšok :sekundy 60  
        ;Prebytočné sekundy sme presunuli do minút. V premennej "sekundy si  
        ;uložíme, koľko sekúnd uplynulo z danej minúty
```

ak2 :sekundy < 10

[hodiny'nechHodnota (slovo :minúty ":0 :sekundy)]

[hodiny'nechHodnota (slovo :minúty ": :sekundy)]

;Nastavíme text hodiny. Ak je počet sekúnd jednociferný, za oddeľujúcu

;dvojbodku pridáme ešte nulu, takže to bude písať napr 1:08 a nie 1:8.

;Príkaz slovo zlepí zadané reťazce dohromady.

ak prázdny? (veta :asteroidy :zničené :výbuchy)

[

zastav "hlavná

zastav "ukážČas]

;Procedúra ukážČas bude poverená ešte jednou úlohou. Ak sú zoznamy

;asteroidy, zničené a výbuchy prázdne, (a prázdny je aj zoznam, ktorý

;dostaneme ich zlepením príkazom veta), zastavíme procesy, ktoré

;spúšťajú hlavnú procedúru aj nastavujú hodiny. Prvý z nich už

;nepotrebujeme a druhý treba zastaviť, aby sme sa mohli pozrieť,

;za koľko sme to vlastne uhrali.

koniec

Teraz by vám hra mohla fungovať aj s časomierou. Vyskúšajte si to a užite si to. V čase písania lekcie je rekord autora 1 minúta a 21 sekúnd. V budúcej kapitole si povieme, ako uchovať najvyššie skóre a o tom, na čo môže byť dobré, keď projekt obsahuje viacero stránok.

Dvadsať štvrtá kapitola


Korytnačka a stránky

V predošlej kapitole sme zariadili, aby sa nášmu hráčovi odmeral čas, počas ktorého zlikviduje všetky asteroidy. V tejto kapitole hru dovedieme takmer²⁴ do dokonalosti – hráčovi ukážeme, ako dopadol a ak uhral najlepší čas v danej hre, alebo dokonca najlepší čas v histórii, tak ho pochválime.

Najprv si spravíme procedúru na porovnanie dvoch časov. Na vstupe dostane dva časy (napríklad 9:31 a 3:47) a v prípade, že prvý čas je lepší, vráti výsledok 1 a v prípade, že je horší, výsledok bude -1. (V uvedenom prípade to teda bude -1.) Procedúra bude vyzerať takto:

```
viem porovnajČas :a :b
ak :a = :b [vy 0]
    ;Ak sú vstupné časy rovnaké, výsledok bude 0
urobTu "dvojbodkaA prvýVýskyt " : :a
urobTu "dvojbodkaB prvýVýskyt " : :b
    ;Zistíme, na akej pozícii sa v jednotlivých reťazcoch nachádza dvojbodka
urobTu "minutyA prvok (veta 1 :dvojbodkaA - 1) :a
urobTu "minutyB prvok (veta 1 :dvojbodkaB - 1) :b
    ;Do „minútových“ premenných uložíme kus reťazca pred dvojbodkou
urobTu "sekundyA bezPozície (veta 1 :dvojbodkaA) :a
urobTu "sekundyB bezPozície (veta 1 :dvojbodkaB) :b
    ;Do „sekundových“ premenných uložíme kus reťazca za dvojbodkou
ak :minutyA > :minutyB [vy -1 ]
ak :minutyA < :minutyB [vy 1 ]
    ;Pri porovnávaní sú rozhodujúce minúty
ak :sekundyA > :sekundyB [vy -1 ]
ak :sekundyA < :sekundyB [vy 1 ]
    ;Ak sú minúty rovnaké, rozhodneme sa podľa sekúnd
koniec
```

Pristúpme teraz k hlavnému bodu programu tejto kapitoly. Vytvoríme novú stránku. Totiž – na jednej stránke bude prebiehať samotná hra, budú sa po nej preháňať asteroidy a bude vidieť textové pole hodiny. Keby sme na tej istej stránke mali zobrazíť dosiahnuté skóre, museli by sme najprv množstvo objektov skryť, potom ukázať ďalšie kvantum objektov ohlasujúcich skóre. Po začatí novej hry by sme museli poskrývať zase tieto objekty a ukázať pôvodné. Bolo by to nešikovné a neprehľadné. Preto to spravíme tak, že si zriadieme stránky dve. Na jednej sa bude hrať, na druhej bude stále tabuľka s dosiahnutým skóre a medzi týmito dvoma stránkami sa budeme prepínať.

Nová stránka sa vytvára tlačidlom . Na hornej lište sa objaví jej ikonka (doteraz tam bola len jedna stránka, teraz tam budú dve). S jej pomocou sa môžete na stránku kedykoľvek prepnúť. Takže sa na novú stránku prepnete a v príkazovom riadku nastavíte farbu pozadia na čiernu príkazom `nechFarbaPozadia "čierna`. Potom popridávajte 10 textových polí a dve tlačidlá tak, ako to vidíte na obrázku na nasledujúcej strane. Vyberte nejaký pekný font a rozmiestnite ich tak, aby to dobre vyzeralo. Dbajte na to, aby boli názvy jednotlivých textových polí také, ako je uvedené v bublinách na obrázku, bude to dôležité, keď sa budeme odvolávať na ich hodnoty. Tlačidlám vo

²⁴ Žiadny program nie je celkom dokonalý. V každom (a teda aj v tomto) sú chyby. Ich vyhľadanie a opravenie ale necháme na láskavého čitateľa :)

vzhľad zaškrtnite voľbu *plochý typ* a ako návod na kreslenie im napíšete nechFp "zelená text [Nová hra] a nechFp "oranžová text [Koniec].

Možno ste si všimli, že medzi tými textami chýba text2. To je dané tým, že je umiestnený presne na tom istom mieste, ako text1, ale obsahuje nápis *Dosiahol si najlepšie skóre v tejto hre*. Oba nadpisy budú normálne skryté (môžete im v kolónke *vzhľad* rovno vypnúť nastavenie *Vidno*) a ukážu sa iba vtedy, keď to bude aktuálne.



Údaje, s ktorými budeme pracovať, sú uložené v textoch text7, text8, text9 a text10. Časy sú zatiaľ nastavené na 10 minút (predpokladá sa, že za ten čas to uhrá každý) a meno fiktívneho rekordmana je zatiaľ Jozef Rekordér.

Je chyba, že jednotlivým textovým poliam boli ponechané názvy, ktoré im boli pridelené automaticky. Podobne ako je dobré dávať významové názvy premenným, pretože to zvyšuje čitateľnosť programu, je dobré to robiť aj s

inými objektami v Imaginovskom projekte. Ak si niekto trúfa nepomyliť sa, nech si názvy kľudne zmení.

Textové polia text9 a text10 v sebe nesú informáciu o tom, aké je najlepšie skóre v histórii a kto ho uhrá. Tieto údaje treba nejako uchovávať aj v čase, keď je hra vypnutá. Treba ich teda nejako uložiť na disk. Budeme ich zapisovať do súboru *astehiscore.txt*. Procedúra uložiťSkóre bude vyzeráť takto:

viem uložiťSkóre

Stránka2'text10'nechUpravovanie "nie

;Ak bol ešte text10 v režime upravovania, tak sa ten režim vypne

ak Stránka2'text7'hodnota <> Stránka2'text9'hodnota [ukončí]

;Ak je posledné uhraté skóre (text7) iné ako rekord, tak aktuálny rekord
;netreba ukladať znovu, lebo sa budeme starať aby bol uložený hneď po
;dosiahnutí

písanieDo "astehiscore.txt

;Týmto príkazom spôsobíme, že všetky výpisy (spôsobené príkazmi píš
;alebo zobraz – skratka ZO) sa nebudú vypisovať do okna, ale budú sa
;zapisovať do súboru *astehiscore.txt*)

zo Stránka2'text9'hodnota

zo Stránka2'text10'hodnota

;Zapíšeme potrebné hodnoty do súboru

písanieDo []

;Vypneme písanie do súboru, aby sa tam nezapisovali iné hlúposti

koniec

Keď túto procedúru vytvoríte, skúšobne ju spustíte z príkazového riadku. (Predtým si nejako zariadte, aby časy v poli text7 a text9 boli rovnaké.) Ak sa po spustení tejto procedúry pôjdete pozrieť do adresára, v ktorom máte uložený projekt, zistíte, že vám v ňom pribudol súbor *astehiscore.txt*. Otvorte si ho (napríklad notepadom) a pozrite si jeho obsah.

Okrem tejto procedúry budeme potrebovať ďalšiu, ktorú zavoláme pri spustení projektu a ktorá nám údaje zo súboru zase prečíta. Tá môže vyzeráť napríklad takto:

viem čítajSkóre

ak nieJe súbor? "astehiscore.txt

[Stránka2'text9'nechHodnota "|10:00|

Stránka2'text10'nechHodnota "|Jozef Rekordér|

ukonči]

;Najprv si skontrolujeme, či súbor *astehiscore.txt* vôbec existuje. Ak ešte
;neexistuje (lebo sme ešte hodnoty nezapisovali) textové polia *text9* a
;*text10* sa nastaví na preddefinované hodnoty.

čítanieZ "astehiscore.txt

;Nastavíme, že príkaz čítajSlovo nebude čítať z príkazového riadku, ale
;súboru *astehiscore.txt*

zo

Stránka2'text9'nechHodnota čítajSlovo

Stránka2'text10'nechHodnota čítajSlovo

;Hodnoty textových polí nastavíme podľa toho, ako sme sa dočítali
;v súbore. Čítať musíme v rovnakom poradí, v akom sme zapisovali, inak
;by sa do poľa *text9* dostalo nejaké meno a do poľa *text10* nejaký čas.

čítanieZ []

;Znovu prepne čítanie na príkazový riadok.

koniec

Dobre. Procedúry na zápis a čítanie z disku sme zvládli. Teraz spravíme jednu väčšiu procedúru, ktorá nám upraví stránku č.2 po tom, ako hráč dohrá partiu. Budeme v nej hojne využívať procedúru porovnajČas, ktorú sme si spravili na začiatku kapitoly.

viem skóre

Stránka2

;Prepneme sa na stránku 2

ak2 (porovnajČas Stránka1'hodiny'hodnota Stránka2'text9'hodnota) = 1

[

Stránka2'text1'ukážMa

Stránka2'text2'skryMa

Stránka2'text7'nechHodnota Stránka1'hodiny'hodnota

Stránka2'text8'nechHodnota Stránka1'hodiny'hodnota

Stránka2'text9'nechHodnota Stránka1'hodiny'hodnota

Stránka2'text10'nechHodnota " | |

Stránka2'text10'nechUpravovanie "ano

]

;Ak je dosiahnutý čas lepší, než doterajší rekord, ukážeme pochvalu
;nastavíme čas na hodinách do polí *text7*, *text8* a *text9* (dosiahnutý čas,
;dnešný a absolútny rekord), vymažeme starého rekordmana a pole
;nastavíme na upravovanie, nech sa tam môže dopísať nový.

text1,

text10

[

ak2 (porovnajČas Stránka1'hodiny'hodnota Stránka2'text8'hodnota) = 1

[

```

Stránka2'text1'skryMa
Stránka2'text2'ukážMa
Stránka2'text7'nechHodnota Stránka1'hodiny'hodnota
Stránka2'text8'nechHodnota Stránka1'hodiny'hodnota
]
;Ak sme prekonalí dnešný rekord, skryje sa pochvala č.1, ukáže sa
;pochvala č.2 a stav hodín sa prepíše do polí text7 a text8.
[
Stránka2'text1'skryMa
Stránka2'text2'skryMa
Stránka2'text7'nechHodnota Stránka1'hodiny'hodnota
]
;Ak sme nič neprekonali, skryjú sa pochvaly a stav hodín sa prepíše do
;poľa text7.
]
koniec

```

Teraz treba upraviť procedúru `ukážČas`, aby v prípade, že je hra dohnaná zavolala procedúru `skóre`. Dajte si pozor, aby ste volanie procedúry umiestnili ešte pred príkaz `zastav "ukážčas`, pretože keď sa tento príkaz vykoná, zastaví sa aj beh procedúry `ukážČas` a skóre sa už neupraví.

Ďalšia zmena sa bude týkať procedúry `štart`. Musíme zabezpečiť dve veci. Jednak to, že sa hra začne na stránke č.1. Ak by sme totiž spustili procedúru `štart` na druhej stránke, asteroidy by sa nám potulovali pomedzi nadpisy a vyzeralo by to neštýlovo. Spravíme to tak, že celú procedúru `štart` spustíme ako proces hlavného okna (o tom je to `hlavnéOkno'odštartuj`) a na začiatku procesu prepne hlavné okno na stránku 1:

```

viem štart
hlavnéOkno'odštartuj
[
Stránka1
....
]
koniec

```

Miesto tých bodiek tam samozrejme bude celá pôvodná procedúra `štart`. Druhá vec, ktorá sa v procedúre `štart` musí udiť, je nastavenie skóre na stránke č.2. Takže pred spustenie procesu hlavnej procedúry pridajte tieto dva riadky:

```

čítajSkóre
Stránka2'text8'nechHodnota "|10:00|

```

Procedúra `čítajSkóre` prečíta z disku najlepšie historické skóre. Druhý príkaz nastaví dnešné najlepšie skóre na štandardných desať minút.

Posledná vec, ktorú musíme spraviť, je obsluha tlačidiel `t1` a `t2`. Najprv sa budeme zaoberať tlačidlom `t2` – Nová hra. Vytvoríme mu procedúru (súkromnú pre to tlačidlo) `nováHra`, ktorú budeme volať pri zapnutí (v základných nastaveniach tlačidla teda do kolónky *priZapnutí* napíšete `nováHra`). Procedúra `nováHra` sa od procedúry `štart` líši len v detailoch (nepotrebuje načítavať najlepšie skóre, zato sa musí postarať o to, aby bolo nové najlepšie skóre uložené) takže vám ju predkladáme bez komentárov:


```

viem nováHra
  hlavnéOkno'odštartuj
  [
    uložSkóre
    Stránka1
    bezPV
    nechFarbaPozadia 0
    zrušObjekt všetky
    urob "asteroidy []
    urob "strely []
    urob "výbuchy []
    urob "zničené []
    nová "raketa []
    nový "asteroid []
    (každých 20 [hlavná] "hlavná)
    vynulujČasomer
    (každých 500 [ukážČas] "ukážčas)
  ]
koniec

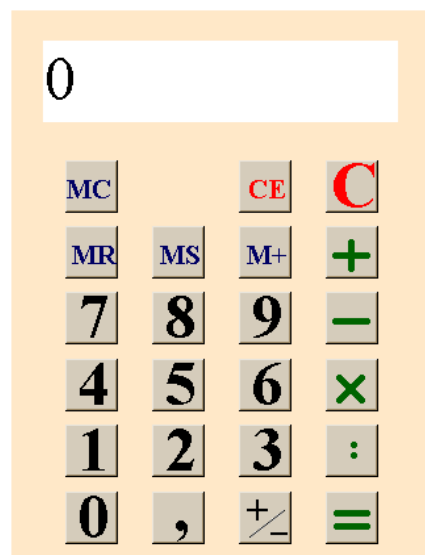
```

Tlačidlo t1 má na starosti iba uloženie nového skóre a zavretie Imaginu, takže mu do kolónky *priZapnutí* napíšete iba uložSkóre bye (príkaz bye zavrie Imagine). Hra je hotová, príjemnú zábavu. Mimochodom – autorov rekord v čase písania týchto riadkov je 1:17.

Dvadsať piata kapitola

Korytnačka a kalkulačka

Kedysi dávno v deviatej kapitole sme hovorili o tom, že korytnačka je múdra a inteligentná a vie celkom dobre počítať. Ale naučiť starú mamu programovať v Imagine len preto, aby si mohla počítať, či ju pri nákupe neošmekli, môže byť celkom problém. Určite by jej oveľa viac prospela bežná kalkulačka. Budeme ignorovať fakt, že každý slušnejší systém (MS Windows nevynechajúc) obsahuje kalkulačku ako štandardnú súčasť a skúsime nejakú kalkulačku naprogramovať. Takže: najprv si vyrobíme papier a na ten vyrobíme a poukladáme tlačidlá a textové pole tak, ako to vidíme na obrázku. Papier sa bude volať `papier1`, tlačidlá pomenujeme podľa toho, čo je na nich napísané, takže ich mená budú 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, bodka, c, ce, delno, krat, m+, mc, minus, mr, ms, plus, plusminus a rovnasa. Textové pole nazveme originálne displej. Vpravo na displeji pridáme ešte jedno textové pole (ktoré na našom obrázku nie je vidno), vložíme do neho jediné písmenko M, nazveme ho `memory` a v kolónke *vzhľad*, mu vypneme možnosť *vidno*. (Áno, to je presne ten dôvod, kvôli ktorému ho na našom obrázku nevidno.) Budeme ho zviditeľňovať a zneviditeľňovať podľa toho, či práve máme niečo uložené v pamäti.



Okrem tlačidiel a displeja (ktorého hodnota nám bude slúžiť ako jedna z premenných) si potrebujeme zriadiť ešte nejaké ďalšie premenné. Môžete si ich vyrobiť z príkazového riadku príkazom `urob` alebo priamo v okne *pamät*. Prvá z nich bude `starecisko`. Bude dobrá na nasledujúcu vec: Keď na kalkulačke rátame napríklad $2 + 3$, tak najprv musíme zadať dvojku, potom operáciu, ktorú chceme urobiť a potom trojku. Keď to dozadáme, trojka je stále na displeji, tá dvojka je ale už zmazaná. Aby sme ale mohli súčet vyrátať, musíme si ju niekde pamätať. A na to budeme používať práve premennú `starecisko`. A samozrejme, musíme si pamätať aj zvolenú operáciu a na to budeme mať premennú `operacia`. Na začiatku v nej bude hodnota nič.

Ďalšia premenná sa bude volať `hotovecisko` a bude obsahovať buď `áno` alebo `nie`. V tejto premennej si budeme pamätať, či je číslo na displeji hotové, alebo ešte nie. Rozdiel je totiž v tom, že ak zadáme kalkulačke $2 + 3 =$ a na displeji sa objaví `5`, tak táto päťka je hotová. Ak teraz stlačíme tlačidlo `1`, tak päťka zmizne a objaví sa jednička, ktorá bude patriť k ďalšiemu výpočtu. Ak ale výpočet začneme tým, že stlačíme tlačidlo `5`, táto päťka hotová nie je. Ak po nej stlačíme tlačidlo `1`, na displeji bude `51`.

No a posledná premenná, ktorú budeme používať, je premenná `pamat`. Budeme do nej ukladať to, čo bude mať kalkulačka v pamäti.

Tlačidlá máme, premenné máme, teraz ešte treba urobiť nejaké procedúry, aby to celé fungovalo. Začneme procedúrou `ciselko`, ktorá sa bude volať, keď niekto stlačí nejakú číslicu:

```
viem ciselko :a
  ak :hotovecisko
  [
    urob "hotovecisko "nie
    displej'nechHodnota :a
  ukonci
```

```

]
;Ak bolo číslo na displeji hotové, tak ho zrušíme a miesto neho tam dáme
;novú číslicu a poznačíme si, že číslo hotové nie je.

```

```

ak zaroven (displej'Hodnota = 0) (nieJe prvok? ". displej'Hodnota)

```

```

[
  urob "hotovecisko "nie
  displej'nechHodnota :a
  ukonci

```

```

]
;Ak je na displeji 0 a nebola spustená desatinná čiarka, tak obsah displeja
;prepíšeme novou cifrou (aby sa nedali vyrobiť paškviľy typu 007)
displej'nechHodnota (slovo displej'Hodnota :a)
;V ostatných prípadoch pripojíme (s pomocou príkazu slovo) číslicu
;na koniec displeja.

```

koniec

Jednotlivým tlačidlám s číslicami ešte nastavíme, aby pri stlačení procedúru `ciselko` zavolali. Teda napríklad na tlačidlo 1 bude v kolónke *priZapnutí* príkaz `ciselko 1`.

Ďalšiu procedúru, ktorá bude obhospodarovať stlačenie desatinnej čiarky, nazveme rafinovane `bodka`. Bude vyzeráť takto:

```

viem bodka
  ak :hotovecisko
  [ displej'nechHodnota "0.
    urob "hotovecisko "nie
    ukonci ]
;Ak je na displeji hotové číslo, zrušíme ho a dáme tam „0.“
ak prvok? ". displej'hodnota
[ukonci]
;Ak už tam desatinná čiarka je, nerobíme nič.
displej'nechHodnota (slovo displej'hodnota ".)
;V ostatných prípadoch na koniec pripojíme bodku.

```

koniec

Hlavnou procedúrou, ktorá sa bude starať o výpočty, bude procedúra `vyhodnot`. Bude mať na starosti pozrieť, čo je v starom čísle, aká operácia bola zvolená a spraviť patričnú operáciu. Výsledok hodí aj na displej, aj do premennej `starecisko`. Bude vyzeráť takto:

```

viem vyhodnot
  ak2 :operacia = "nic
  [ urob "starecisko displej'hodnota ]
;Ak nie je žiadna operácia, tak sa toho moc nedeje
[
  akje :operacia
  [
    plus [ urob "starecisko :starecisko + displej'hodnota ]
    minus [ urob "starecisko :starecisko - displej'hodnota ]
    krat [ urob "starecisko :starecisko * displej'hodnota ]
    deleno [ urob "starecisko :starecisko / displej'hodnota ]

```

```

]
;Ak je operácia určená, podľa nej sa prevedie patričný výpočet
]
displej'nechHodnota :starecisko
urob "hotovecisko "ano
;Výsledok hodíme na displej a označíme ho ako hotový
koniec

```

Môžeme rovno povedať, čo sa má stať pri stlačení =. Musíme zavolať vyhodnotenie a súčasne nastaviť, že momentálne žiadnu operáciu nerobíme. Do kolónky *priZapnutí* je teda treba dopísať

```
vyhodnot urob "operacia "nic
```

Z procedúr nám chýba spraviť už len najjednoduchšiu – procedúru *operacia*. Ak stlačíte nejakú operáciu, treba vyhodnotiť to, čo bolo predtým a operáciu si zapamätať. Takže naša procedúra bude vyzeráť takto:

```
viem operacia :aka
vyhodnot
urob "operacia :aka
koniec

```

Teraz treba na tlačidlo plus zavesiť príkaz *operacia "plus*, na tlačidlo minus príkaz *operacia "minus*, na tlačidlo krát príkaz *operacia "krat* a na tlačidlo deleno príkaz *operacia "deleno*.

Teraz nám ostáva už len sfunkčoniť ostatné tlačidlá. To pôjde celkom jednoducho.

- Na tlačidlo *plusminus* zavesíme príkaz

```
displej'nechHodnota -1 * displej'hodnota
```

 (keď niečo vynásobíme mínus jednotkou, zmeníme tomu znamienko).
- Na tlačidlo *ce* zavesíme príkaz *displej'nechHodnota 0*
- Na tlačidlo *c* zavesíme príkaz

```
displej'nechHodnota 0 urob "starecisko 0 urob "operacia "nic
```

 (Musíme vynulovať úplne všetko. Aj staré číslo a operáciu.)
- Na tlačidlo *ms* zavesíme príkaz

```
urob "pamat displej'hodnota memory'nechVidno ( :pamat <> 0 )
```

 (Do pamäte vložíme hodnotu displeja. Ak je to, čo je v pamäti, nenulové, rozsvieti sa M. Ak je to nula, M zhasne.)
- Na tlačidlo *m+* zavesíme príkaz

```
urob "pamat : pamat + displej'hodnota memory'nechVidno ( :pamat <> 0 )
```

 (To, čo je na displeji, pripočítame k pamäti. Zas podľa toho, čo je v pamäti, rozsvietime, alebo zhasneme M.)
- Na tlačidlo *mr* zavesíme príkaz

```
displej'nechHodnota :pamat urob "hotovecisko "ano
```

 (To, čo je v pamäti, hodíme na displej a zapamätáme si, že je to hotové.)
- Na tlačidlo *mc* zavesíme príkaz

```
urob "pamat 0 memory'nechVidno "nie
```

 (Vynulujeme pamäť a zhasneme M)

No a to je všetko. Môžete kalkulačkovať, koľko len chcete.