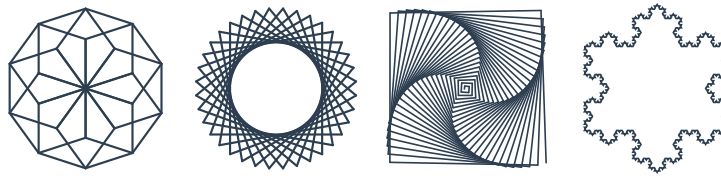
Želví grafika

Exkurze do programování, geometrie a umění

Radek Pelánek



Umíme informatiku

Radek Pelánek, červenec 2022.

Pro snadnější využití knihy ve výuce jsou na následující adrese k dispozici vybrané obrázky z knihy:
https://www.umimeinformatiku.cz/metodika-zelvi-grafika-obrazky

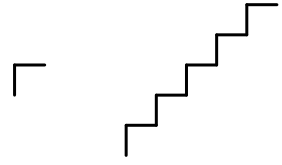
Tato knížka vyšla původně tiskem v roce 2018. V mezičase od publikování původní verze jsme cvičení pro želví grafiku zpracovali v rámci systému Umíme informatiku (https://www.umimeinformatiku.cz/). Současná verze je doplněná především o popis těchto cvičení (Příloha A).

Obsah

1 Jak jsem potkal želvu

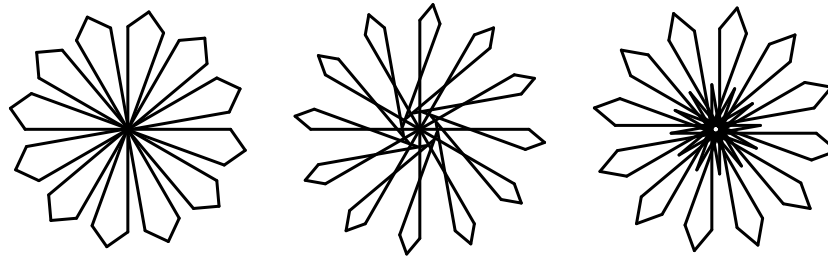
Pořídil jsem si želvu. Jmenuje se Žofka a není to vůbec obyčejná želva. Brzy jsem si všiml, že umí lozit krásně rovně a zanechává za sebou v písku pěkné čáry. Navíc je velmi inteligentní, takže rozumí různým příkazům. A také se ráda učí nové věci.

Tak jsme spolu začali experimentovat – dali jsme se na kreslení obrázků. Místo do písku kreslíme na papír. Přivázal jsem jí opatrně na ocásek malé pero, kterým dělá čáry. Takže když jí řeknu „běž dopředu, otoč se doprava, běž dopředu, otoč se doleva“, vykreslí dvě kolmé čáry. A když tyto příkazy pětkrát zopakuji, vykreslí schody:



Budu vám vyprávět o tom, jak jsme se s Žofkou učili kreslit čím dál složitější obrázky. Naučili jsme se při tom spoustu věcí o programování, matematice a taky o umění.

Pokusy se želvou byly velmi zajímavé, mimo jiné proto, že želva chyby netrestá, ale odměňuje. Často se mi stalo, že jsem chtěl s pomocí želvy vykreslit nějaký obrázek, ale pokyny, které jsem jí dal, nebyly úplně správně. Žofka vždy obrázek trpělivě vykreslila a výsledek mě mnohdy překvapil – občas byl i hezčí než můj původní záměr. Tak třeba tyto obrázky:



Původně jsem chtěl vykreslit ten první – jednoduchou kytičku. Špatně jsem si ale vypočítal vzdálenosti a úhly, jaké mám želvě zadat, a Žofka tak vykreslila při mých pokusech další dva obrázky, které se mi nakonec líbí víc.

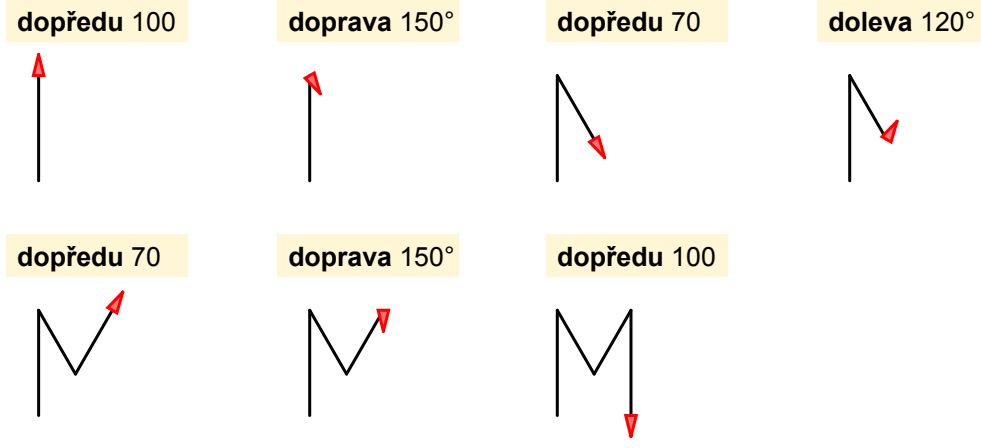
Chtěli byste si vyzkoušet kreslení s námi? Pokud zrovna nemáte po ruce velmi inteligentní a trpělivou želvu, nezoufejte. Želví grafiku si můžete zkusit i na počítači. V příloze na konci knihy najdete přehled možností a rady k jejich použití.

Abyste měli impulz k vlastnímu experimentování, neprozradím vám u všech obrázků, jak přesně jsme je s Žofkou vykreslili, a nechám vám je jako výzvy k samostatnému řešení. Tyto výzvy jsou uvedeny vždy na konci kapitoly a pomocí hvězdiček je vyznačena jejich obtížnost. Neváhejte vymýšlet i další, vlastní výzvy. Když se naučíte přemýšlet jako želva, najdete kolem sebe spoustu zajímavé inspirace.

Pojďte s námi na překvapivě dobrodružnou výpravu, při které potkáme zajímavou společnost zdánlivě nesouvisejících obrázků a pojmů z programování, matematiky a umění: hvězdy, diamanty, Ulamovu spirálu, křivku srdcovku, květ života, Fibonačiho strom, Kochovu vločku, Sierpińského trojúhelník, fraktální kytičky, keltské ornamenty, indické rekurzivní umění i prošívané deky.

2 Želva se učí kreslit

Nejprve jsme s Žofkou museli zvládnout základní příkazy. Ty úplně nejdůležitější jsem již zmiňoval: posunutí dopředu a otočení doprava či doleva. Aby Žofka přesně věděla, co má dělat, nestačí však jen příkaz „jdi dopředu“ nebo „zatoč doprava“. Musím jí také říct, o kolik přesně se má posunout a jak moc má zatáčet. Takže příkaz k přesunu vždy doprovázím vzdáleností, která udává délku posunu, a příkaz k otočení doprovázím úhlem, o který se má želva otočit. Vypadá to třeba takto:

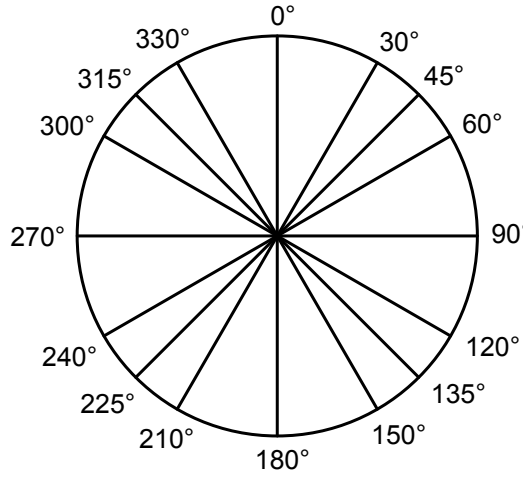


„A ještě mi můžeš dávat příkaz pro couvání dozadu, to taky zvládnu,“ navrhla Žofka.

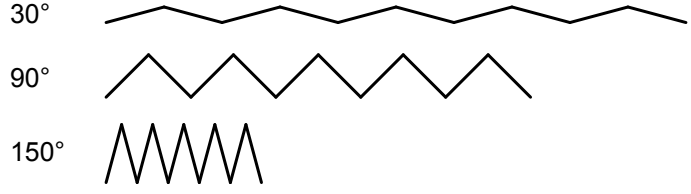
„Já vím, že jsi šikovná želva a umíš couvat. To je dobře. Ale speciální příkaz pro couvání potřebovat nebudeme. Místo toho tě naučím používat záporná čísla. Posun dopředu o záporné číslo je totiž couvání, například **dopředu** -100 znamená to stejné, jako by znamenalo **dozadu** 100,“ vysvětlil jsem jí.

Úhly

Žofka je sice hodně chytrá, ale pro začátek jsem jí musel vysvětlit, co znamenají úhly a jak je měříme: Že celá otočka je $360°$, otočka na druhou stranu je $180°$, pravý úhel je $90°$ a tak dále. Protože to na začátku trochu pletla, připravil jsem jí návodný obrázek. Do něj jsem vyznačil úhly, které jsme nejčastěji používali:



Jako rozcvičku pro procvičení úhlů jsme zkusili vykreslit jednoduchou klikatou čáru, u které se střídá zatáčení doleva a doprava o zadaný úhel:



Barvy, šířky, tečky

Poté, co jsme si natrénovali základní kreslení, začala Žofka trochu vymýšlet: „Já bych chtěla kreslit i pestřejší obrázky, nejen pořád samé stejné černé čáry."

A tak jsem Žofku postupně naučil další příkazy. Vcelku jednoduché jsou příkazy „zvedni pero" a „polož pero" – Žofka podle nich zvedá ocásek a s ním i pero. Při kreslení za sebou nechává čáru, jen pokud má položené pero. S pomocí těchto příkazů tak může kreslit třeba přerušovanou čáru.

„A jak uděláme barvy?“ zajímala se Žofka.

„Co kdybych ti na ocas přidělal několik barevných per? Ty ho vždy natočíš tak, abys kreslila správnou barvou,“ navrhl jsem.

„Obávám se, že přeceňuješ schopnosti mého ocásku,“ odvětila Žofka.

Tak jsem jí pořídil pomocníka – trpaslíka Tondu. Tonda jezdí Žofce na krunýři, má batoh plný barevných per a pomáhá s následujícími příkazy. Na příkazy „nastav barvu“ a „nastav tloušťku“ Tonda vymění želvě na ocásku pero tak, aby mělo správnou barvu a tloušťku. Na příkaz „udělej tečku“ Tonda udělá vybarvenou tečku zadané velikosti na místě, kde má zrovna želva ocásek. Díky těmto novým příkazům Žofka s Tondou zvládnou třeba následující obrázky:

```
dopředu 40
zvedni pero
dopředu 20
polož pero
dopředu 40
```

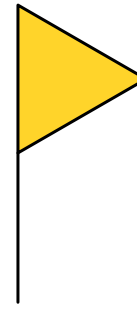
```
nastav barvu modrá
dopředu 70
nastav barvu červená
nastav tloušťku 3
dopředu 30
```

```
dopředu 50
zvedni pero
udělej tečku
dopředu 50
udělej tečku
```


Vyplňování

Nakonec jsme natrénovali příkazy pro vybarvování: „začátek vyplňování“ a „konec vyplňování“. Na pokyn k ukončení vyplňování Tonda seskočí z želvy a temperovou barvou pěkně vymaluje celou oblast, kterou želva od začátku vyplňování obkroužila. Vypadá to třeba takto:

```
dopředu 50
začátek vyplňování žlutě
dopředu 50
doprava 120°
dopředu 50
doprava 120°
dopředu 50
konec vyplňování
```



Když používáme vyplňování, musíme myslet na to, že pořadí vykreslování je důležité. Pokud vykreslíme vyplněné oblasti, které se překrývají,

bude na konci vidět jen poslední vykreslená vrstva (Tonda používá kvalitní temperové barvy). To nám může často usnadnit práci.

Například při vykreslování české vlajky nemusíme vykreslovat přímo červený lichoběžník, který na vlajce vidíme. Stačí vykreslit obyčejný obdélník a přes něj pak vykreslit modrý trojúhelník.

Výzvy

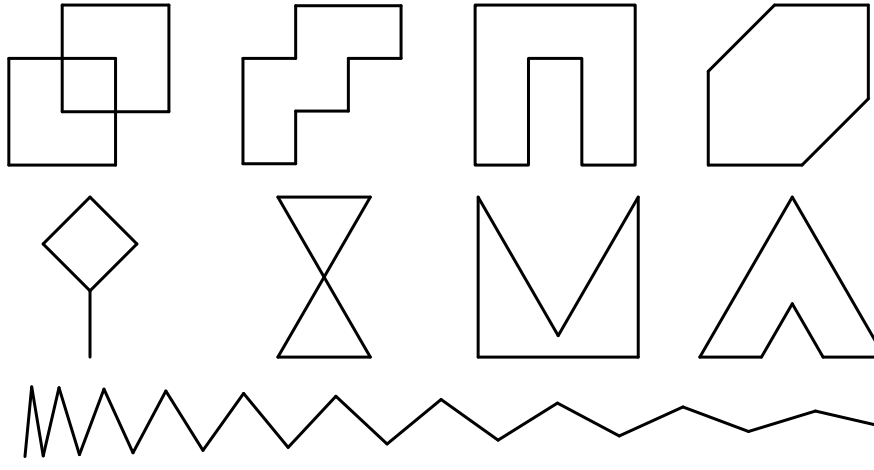
Základy na Umíme informatiku ⋆

Vyřešte úlohy ze sady Základy v systému Umíme informatiku (v blokovém programování nebo v Pythonu):

`https://www.umimeinformatiku.cz/zelvi-grafika`
`https://www.umimeinformatiku.cz/python-zelva`

Jednoduché tvary ⋆

Pomocí základních příkazů (dopředu, doleva, doprava) vykreslete následující tvary:



Želví abeceda ⋆

Jaká písmena vykreslí následující posloupnosti příkazů?

1. **dopředu** 100, **doleva** 90°, **dopředu** 40, **doleva** 180°, **dopředu** 80
2. **dopředu** 100, **doprava** 135°, **dopředu** 140, **doleva** 135°, **dopředu** 100
3. **dopředu** 100, **doleva** 180°, **dopředu** 50, **doleva** 45°, **dopředu** 70, **doleva** 180°, **dopředu** 70, **doprava** 90°, **dopředu** 70

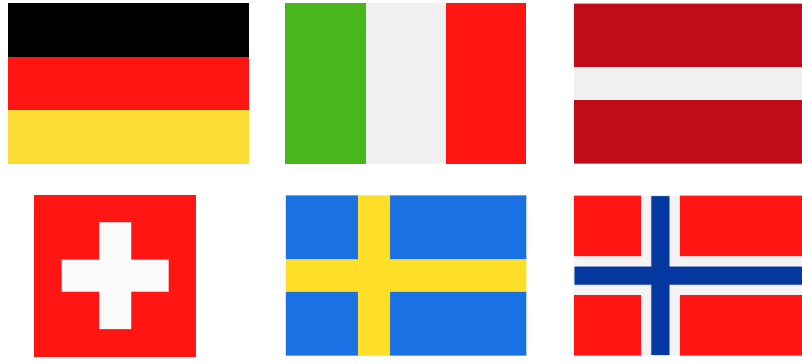
Napište programy pro vykreslení dalších písmen.

Morseova abeceda ⋆

Morseova abeceda kóduje písmena pomocí teček a čárek. Například A je „· −“, B je „− · · ·“. Naučte želvu vykreslovat vybraná písmena v Morseově abecedě.

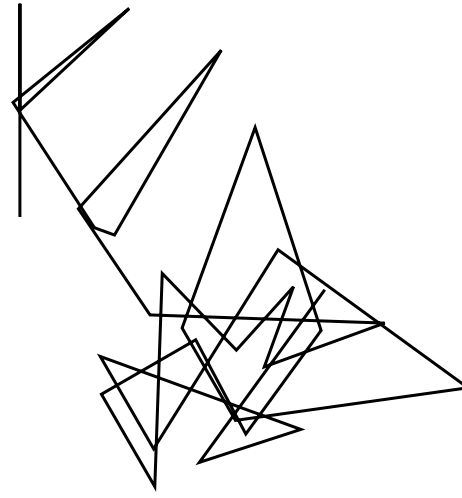
Vlajky ⋆⋆

Vykreslujte vlajky dalších zemí. Jednoduché jsou ty, které jsou jen z pruhů (Německo, Itálie, Lotyšsko). Zajímavější, ale stále zvládnutelné pomocí jednoduchých příkazů jsou ty s křížem (Švýcarsko, Švédsko, Norsko).



Šifra opilé želvy ⋆⋆

Vyluštěte následující nápis, který nakreslila Žofka, když se jednou omylem místo vody napila vodky. Byla pak trochu opilá a při kreslení zatáčela čím dál tím víc doprava (při každé zatáčce se zatočení navíc zvětšovalo o 2 stupně).

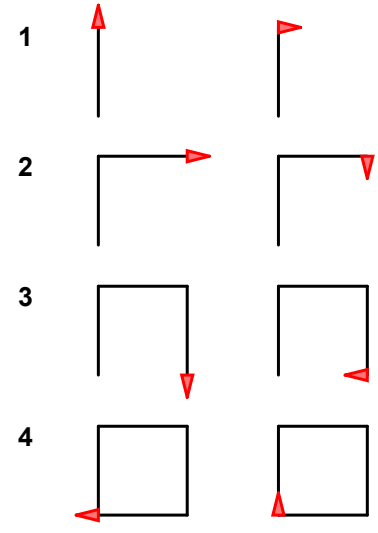


3 Želva opakuje

Doposud jsme vykreslovali obrázky „čáru po čáře“. To nám posloužilo pro natrénování základních příkazů a na procvičení úhlů, ale na vykreslování opravdu pěkných obrázků to zatím nebylo – abychom nakreslili zajímavý obrázek, musel bych želvě dát hrozně moc příkazů. Naučil jsem tedy Žofku opakovaně provádět zadaný blok příkazů. Ukázalo se, že i s tímto jednoduchým přístupem už dokážeme vyrábět zajímavé obrázky.

Základní myšlenka „opakování“ je jednoduchá. Následující obrázek ukazuje, jak želva kreslí čtverec – jde dopředu, zahne o 90 stupňů a takto dokola čtyřikrát. Místo toho, abych příkazy **dopředu** a **doprava** zadával čtyřikrát za sebou, využiji příkaz **opakuj**.

```
dopředu 100
doprava 90°
dopředu 100
doprava 90°
dopředu 100
doprava 90°
dopředu 100
doprava 90°
```

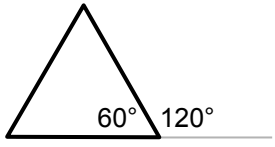



```
opakuj 4×
    dopředu 100
    doprava 90°
```

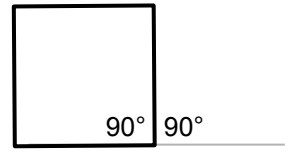
Mnohoúhelníky

Když jsme zvládli čtverec a opakování, zkusili jsme další mnohoúhelníky. Pro vykreslování N-úhelníku je zřejmé, že potřebujeme opakovat N-krát vykreslení čáry a zatočení. „O jaký úhel mám zatáčet?“ potřebovala pochopitelně vědět Žofka. Pro některé mnohoúhelníky je to celkem jasné:

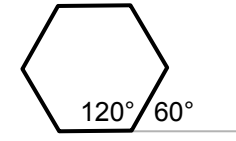
```
opakuj 3×
   dopředu 100
   doleva 120°
```




```
opakuj 4×
   dopředu 100
   doleva 90°
```




```
opakuj 6×
   dopředu 100
   doleva 60°
```



Ale co třeba pětiúhelník? U něj není na první pohled patrné, jaký je správný úhel. S trochou experimentování jsme s Žofkou dospěli k zatáčení o $72°$. Ale přístup „s trochou experimentování“ nám příliš nepomůže vykreslit obecný mnohoúhelník.

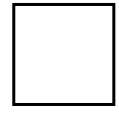
Naštěstí Žofka přišla se zajímavým pozorováním: „Když udělám cestu, při které na konci stojím natočená stejným směrem jako na začátku, tak součet zatáček po cestě musí dávat $360°$!“

Hnidopišsky jsem ji opravil: „Nemáš úplně pravdu. Nemusí to být $360°$, ale může to být třeba i $720°$ nebo jiný násobek $360°$.“

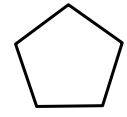
Nicméně musel jsem jí dát za pravdu, že to je užitečné pozorování, které nám pomůže vyřešit náš problém s mnohoúhelníky. Při vykreslování N-úhelníku udělá želva jedno „kolečko“, celkový součet úhlů po cestě tedy musí být $360°$. Celkem udělá N zatáček, všechny zatáčky jsou stejné, takže každá zatáčka musí být o úhel $360°/N$. A máme pěkné obecné řešení!

```
opakuj N×
   dopředu 100
   doleva 360° / N
```

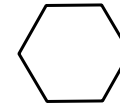
N = 4
90°



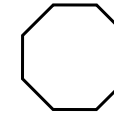
N = 5
72°



N = 6
60°



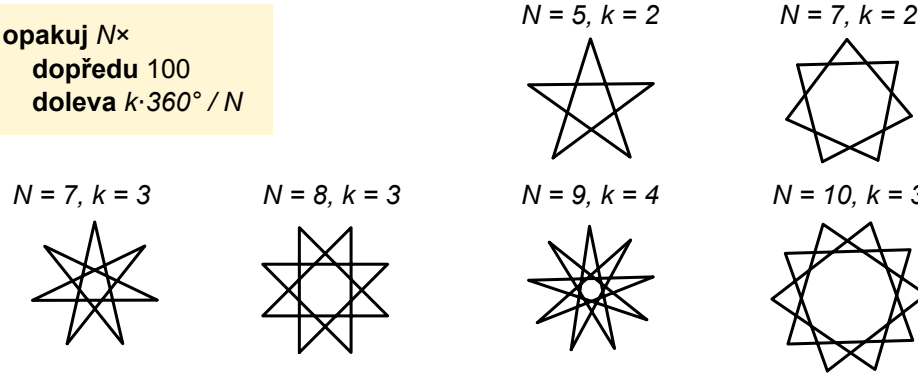
N = 8
45°



Hvězdy

Když jsme zvládli mnohoúhelníky, pustili jsme se do hvězdiček. Základní hvězda s N vrcholy je velmi podobná N-úhelníku. Hvězda má také N vrcholů i hran, jen nespojujeme vždy dva sousední vrcholy, ale „přeskakujeme“. Oproti N-úhelníku tedy potřebujeme pouze upravit úhel, o který želva zatáčí. Na vykreslování hvězdy se můžeme podívat přes Žofčino pozorování o součtu úhlů. Pokud želva při vykreslování „přeskakuje“ vrcholy o k, celkově při vykreslování udělá k „koleček“. Celkový součet úhlů bude $k \cdot 360°$, želva tedy bude zatáčet o $k \cdot 360°/N$.

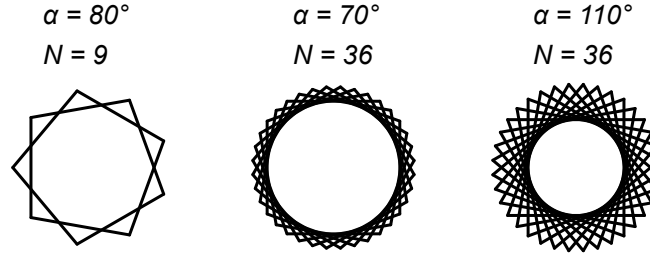
```
opakuj N×
   dopředu 100
   doleva k·360° / N
```



Ale pozor, tento přístup funguje jen tehdy, když N a k jsou nesoudělné, tedy když jejich největší společný dělitel je 1. Pokud bychom třeba pro $N = 8$ zvolili $k = 2$ (přeskakujeme vrcholy ob jedna), nedostaneme žádnou hvězdu, ale obyčejný čtverec.

Ke hvězdám se můžeme dostat i z jiné strany. Řekněme, že budeme opakovaně provádět posun dopředu a zatočení doleva o úhel α. Kolikrát musíme tuto posloupnost zopakovat, aby se útvar uzavřel? Tady máme několik takto vykreslených příkladů se správně určeným počtem opakování:

```
opakuj N×
   dopředu 100
   doleva α
```



„Jak jsi ale určil správné N?“ zajímala se Žofka. „Vypočítal jsem nejmenší společný násobek α a 360. Ten jsem potom podělil α,“ odpověděl jsem. „Cože?“ nechápala Žofka.

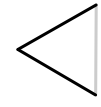
Zkusil jsem jí to tedy osvětlit trochu podrobněji. Pokud se má útvar uzavřít, musí želva na konci skončit ve stejné pozici jako na začátku, což podle dříve zmíněného pozorování znamená, že celkový součet úhlů musí být násobek 360. Celkový součet úhlů tedy musí být současně násobek α a 360. Nás zajímá nejmenší potřebný počet opakování, takže hledáme nejmenší společný násobek. Ten udává celkový součet úhlů, počet opakování N pak dostaneme podělením α. Celkově tedy $N = NSN(\alpha, 360)/\alpha$.

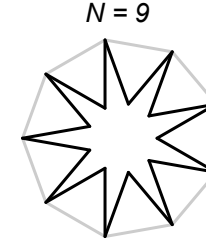
Opakované klikyháky

Zkusme další variaci na mnohoúhelník. Tentokrát necháme úhly stejně jako u základního N-úhelníku (tedy $360°/N$), ale místo prosté rovné čáry uděláme klikyhák. „Klikyhák“ je prostě libovolná posloupnost pohybů a zatáček. Na následujícím obrázku je pro názornost šedě znázorněn N-úhelník, který bychom dostali, pokud by želva místo klikyháku prováděla prostý posun vpřed.

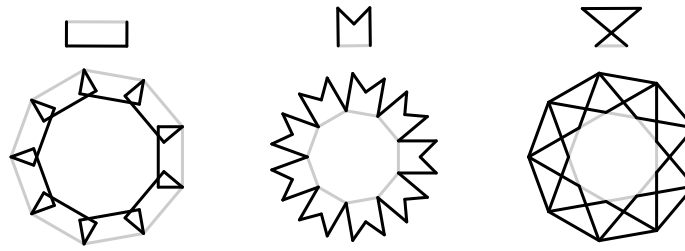
```
opakuj N×
    doleva 50°
    dopředu 30
    doprava 100°
    dopředu 30
    doleva 50°
    doleva 360° / N
```

klikyhák místo rovné čáry

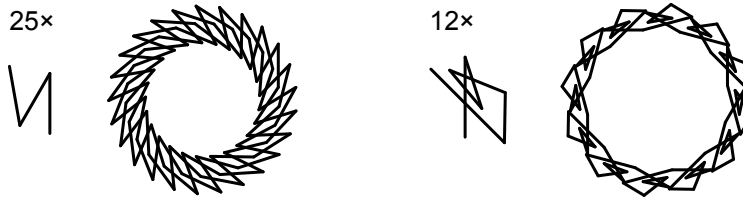




Aby nám vyšel pěkný obrazec, musíme pouze hlídat, aby celkový součet úhlů po cestě byl 0 (nebo násobek 360), přičemž zatáčka doprava o -5° je to stejné jako zatáčka doleva o 5°. Pokud se klikyhák skládá jen z pár symetricky umístěných čar, dostaneme další variace na hvězdy. Zajímavější obrázky dostaneme pro křížící se klikyháky.



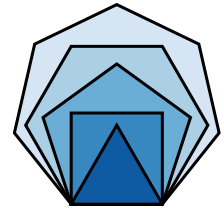
„To je všechno moc uspořádané. Já chci zkusit i nějaký divočejší klikyhák!“ dožadovala se Žofka. A tak jsme si trochu zaexperimentovali. Ukázalo se, že i divoké klikyháky, které samy o sobě nijak zajímavě nevypadají, vytvoří při dostatečném opakování pěkný obrazec.



Výzvy

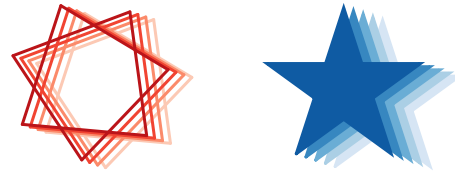
Opakování a vzory na Umíme informatiku ⋆⋆
Vyřešte úlohy na opakování v systému Umíme informatiku. Můžete použít buď blokové programování (sady Jednoduché kreslení, Kreslení, Vzory s opakováním) nebo Python (sady Opakování a Zajímavé vzory):
`https://www.umimeinformatiku.cz/zelvi-grafika`
`https://www.umimeinformatiku.cz/python-zelva`

Barevné mnohoúhelníky ⋆
Nakreslete mnohoúhelníky přes sebe. Pro lepší efekt využijte vybarvování.



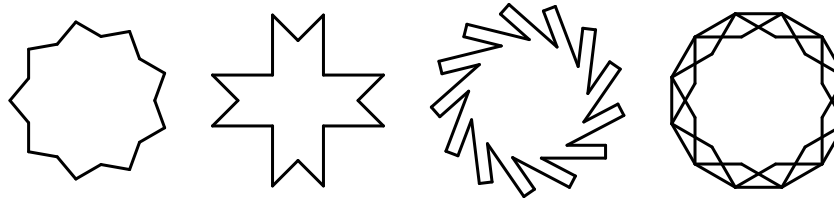
Barevné hvězdy ⋆

Vykreslete barevné hvězdy se „stíny“. Vyzkoušejte jak hvězdy z čar, tak vyplněné.



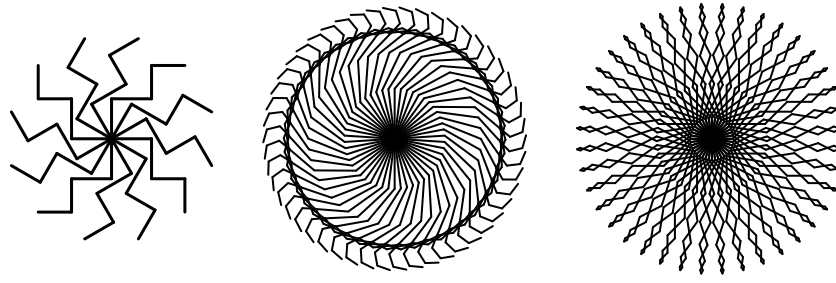
Klikyhákové variace ⋆⋆

Nakreslete následující obrazce (vše jsou to variace na klikyháky).



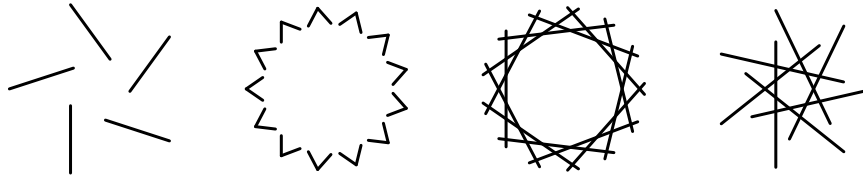
Středové klikyháky ⋆⋆

Jako variaci na „opakované klikyháky“ můžeme zkusit „středové klikyháky“. Tentokrát klikyhák uděláme tak, že začíná a končí na stejné pozici, tj. želva „nacouvá“ zpět na původní místo. Otáčením pak opět dostaneme zajímavé obrazce i z jednoduchých klikyháků.



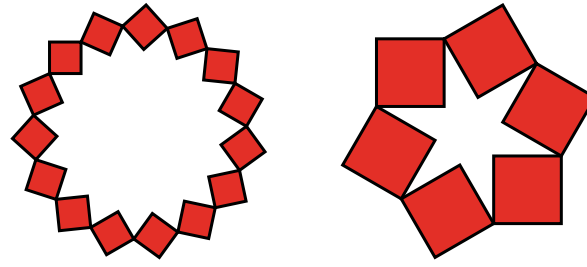
Hvězdy s přerušováním ⋆⋆

Do vykreslování hvězd zkuste přidat zvedání a pokládání pera, tj. místo prostého dopředu proveďte posloupnost pohybů dopředu střídavě se zvednutým a položeným perem. Zkuste tímto způsobem vykreslit například následující obrazce:



Hvězdy z čtverců ⋆⋆

Vykreslete vybarvené čtverce do tvaru mnohoúhelníku a dostanete zajímavou variaci na hvězdy.

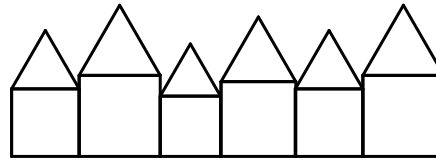


4 Želva se učí nové příkazy

Žofka je nejen inteligentní, ale také velmi učenlivá. Byla schopna nejen provádět základní příkazy, ale také si postupně rozšiřovala slovník příkazů. Když jsem ji naučil něco nového, pojmenovali jsme to jako nový příkaz a mohli dále používat. To se velmi hodilo, protože jinak by vykreslování složitějších obrázků bylo velmi pracné.

Jak se kreslí domečky

Řekněme, že chceme vykreslit řadu domečků vedle sebe:



Na tom není nic moc těžkého. Mohli bychom program prostě vyskládat z dlouhého seznamu posunů dopředu a zatáček. Takový program by ale nebyl pěkný, daleko hezčí řešení dostaneme při využití nově definovaných příkazů.

Žofka nejprve protestovala: „Nechápu, proč se mám učit používat nějaké nové příkazy. Proč bys mi nemohl prostě nadiktovat normální seznam základních příkazů? Nebuď líný!“

„Možnost definovat nové příkazy je v programování velmi důležitá,“ zkoušel jsem ji přesvědčit. „Díky nim jsou programy kratší, přehlednější a elegantnější. A také nám umožňují s programy snadno experimentovat a budovat stále rozsáhlejší programy. Když tě naučím příkaz pro vykreslení domečku, budeme moct velmi snadno kreslit různé ulice, nejen jednu konkrétní řadu domečků.“

Žofka pořád moc nechápala, o čem mluvím, tak jsem jí použití nových příkazů názorně ukázal na těch domečcích. Nejdříve jsem Žofku naučil nové

příkazy pro vykreslení čtverce a trojúhelníku. S jejich využitím jsme pak poskládali domeček.

```
nauč se čtverec délka
  opakuj 4×
    dopředu délka
    doprava 90°
```

```
nauč se trojúhelník délka
  opakuj 3×
    dopředu délka
    doprava 120°
```

```
nauč se domeček délka
  čtverec délka
  dopředu délka
  doprava 30°
  trojúhelník délka
  doleva 30°
  dopředu -délka
```

Pomocí nového příkazu domeček jsme pak již snadno vykreslili řadu domečků. Stejně jako základní příkazy pro pohyb, i nové příkazy mohou mít parametry, které ovlivňují, co přesně příkaz znamená. V tomto případě je parametrem *délka*.

„Není mi jasné, proč je u definice domečku poslední příkaz **dopředu** *-délka*," zajímala se ještě Žofka. „Chápu, že jít dopředu o zápornou délku znamená couvat. Ale není mi jasné, proč tam ten příkaz je – vždyť tímto příkazem stejně jen vykreslím čáru, která už je nakreslená."

Zkusil jsem jí to vysvětlit: „Máš pravdu, že by tam tento příkaz být nemusel a příkaz domeček by stejně vykreslil správný obrázek. Díky tomu couvání však platí, že po provedení příkazu pro vykreslení domečku jsi ve stejné poloze jako před začátkem vykreslování, tedy v levém spodním rohu domečku. To velmi usnadňuje další použití příkazu, třeba pro vykreslení řady domečků."

V programátorské terminologii se nejčastěji pro označení nových příkazů používá pojem „funkce". Někdy se ještě rozlišuje rozdíl mezi procedurami, které provedou zadanou činnost, a funkcemi, které vypočítají výsledek bez provádění vedlejších efektů. Příkazům v želví grafice odpovídá přesněji pojem „procedura".

Želva si rozšiřuje slovník

Užitečné nové příkazy jsou třeba i jednoduché „zkratky" pro posloupnosti příkazů, které často využíváme. Takto jsem třeba Žofku naučil „skok".

„Ty jsi se zbláznil. Už jsi někdy viděl skákající želvu?" bránila se nejdřív Žofka.

„Však to nemusíš brát doslova. Pro účely kreslení obrázků je skok to stejné jako přesun se zvednutým perem," objasnil jsem.

Tento příkaz jsme hned využili pro definici dalších nových příkazů: „Úhyb", který simuluje úkrok do boku, a „čárkovaná čára". Příkaz pro vykreslení čárkované čáry ilustruje využití více parametrů – první parametr udává délku čáry, druhý počet čárek.

```
nauč se skok délka
  zvedni pero
  dopředu délka
  polož pero
```


```
nauč se úhyb délka
  doprava 90°
  skok délka
  doleva 90°
```

```
nauč se čárkovaná čára délka, N
  opakuj N×
    dopředu (délka/N) / 2
    skok (délka/N) / 2
```

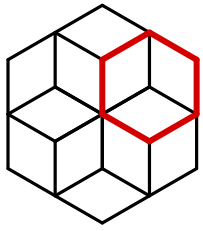
Opakované mnohoúhelníky

Použití nových příkazů jsme prozkoumali na jednoduchém, ale velmi zajímavém příkladu. S využitím dřívějších zkušeností jsem Žofku naučil nový příkaz pro mnohoúhelník. S jeho využitím opakovaně vykreslovala N-úhelníky tak, že všechny mají jeden z vrcholů ve stejném bodě. Při vykreslování Žofka rovnoměrně rotuje, tj. udělá M opakování a mezi dvěma po sobě se otočí vždy o $360/M$ stupňů. V ukázce je pro názornost jeden z mnohoúhelníků zvýrazněn.

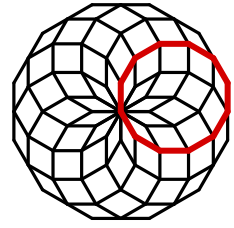
```
nauč se mnohoúhelník N, délka
  opakuj N×
    dopředu délka
    doprava 360° / N
```

```
nauč se diamant N, M, délka
  opakuj M×
    mnohoúhelník N, délka
    doprava 360° / M
```

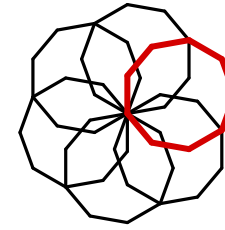
N = 6, M = 6



N = 12, M = 12



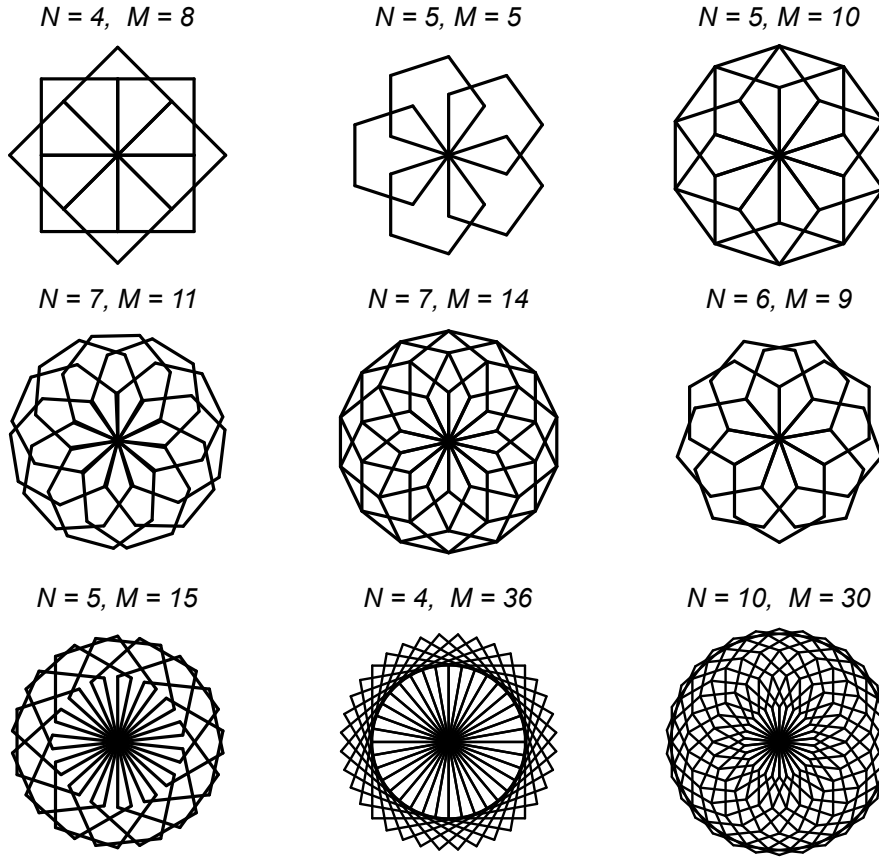
N = 9, M = 6



Z tohoto jednoduchého programu můžeme různou volbou čísel N a M dostat překvapivě pestrou škálu obrázků. Tyto obrázky mohou posloužit jako pěkné omalovánky, ale i jako podnět pro matematické úvahy. Můžeme totiž rozlišit dva základní typy obrázků, a to *diamanty*, u kterých se některé úsečky překrývají (např. $N = 12, M = 12$), a *kytičky*, u nichž se úsečky nepřekrývají (např. $N = 9, M = 6$).

„Pro které volby N, M dostáváme diamanty a pro které kytičky?“ zajímala se Žofka.

„Zkus to vymyslet sama. Dám ti nápovědu: Zamysli se nad významem a vztahem čísel $360°/M$ a $180° - 360°/N$.“



Výzvy

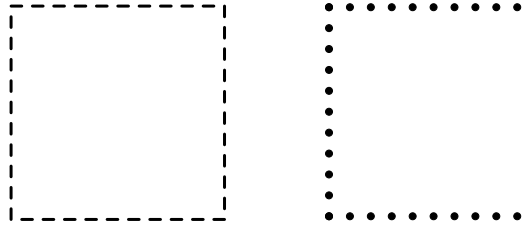
Funkce na Umíme informatiku ⋆

Vyřešit úlohy ze sady Funkce v systému Umíme informatiku (v blokovém programování nebo v Pythonu):

`https://www.umimeinformatiku.cz/zelvi-grafika`
`https://www.umimeinformatiku.cz/python-zelva`

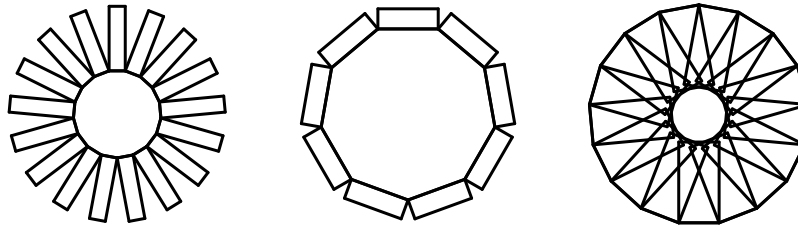
Tečkovaný a čárkovaný čtverec ⋆

Vykreslete čtverce za využití tečkovaných a čárkovaných čar.



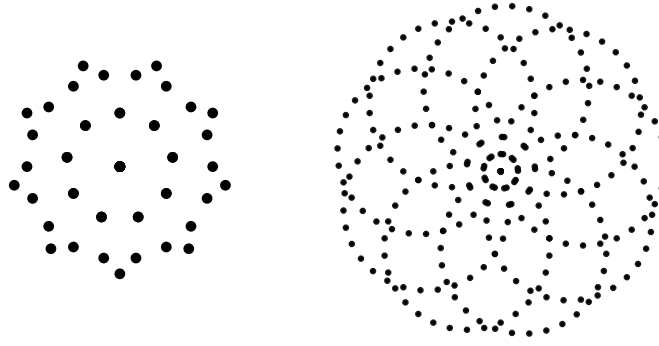
Obdélníky v kolečku ⋆

Opakovaně dokolečka vykreslujte obdélníky. Podle toho, jak nastavíte poměry stran a zda je vykreslíte „dovnitř“ nebo „ven“, dostanete různé obrázky.



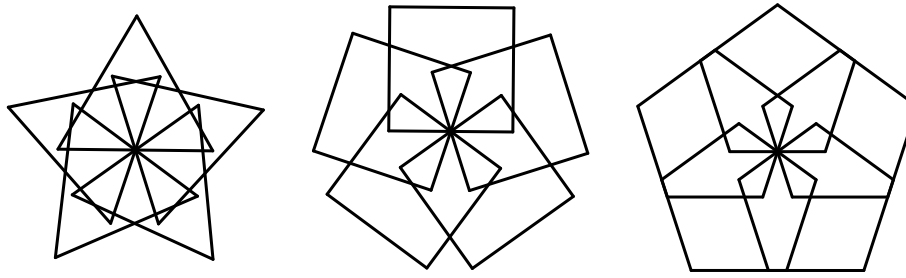
Tečkované opakované mnohoúhelníky ⋆

Upravte příkazy pro vykreslování opakovaných mnohoúhelníků, aby místo čar dělaly jen tečky ve vrcholech.



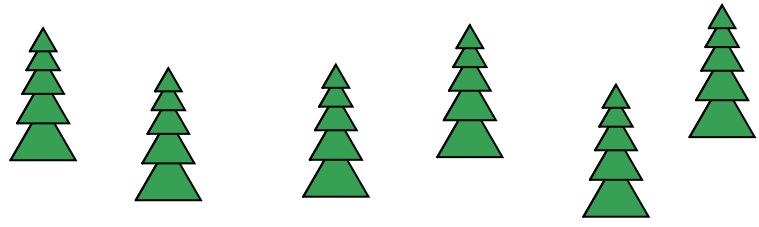
Variace na opakované mnohoúhelníky ⋆

Následující obrázky vzniknou drobnou úpravou programu pro opakované mnohoúhelníky. Zjistěte, v čem je rozdíl, a obrázky vykreslete.



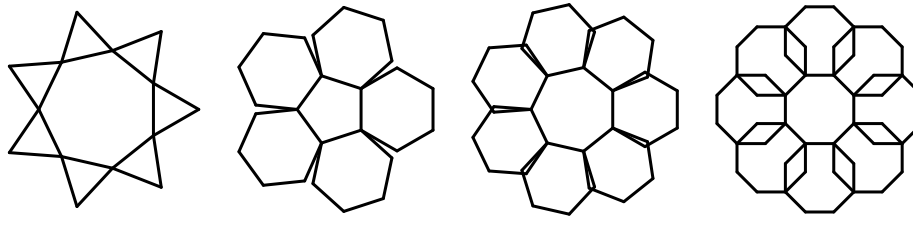
Les ⋆

Z vybarvených trojúhelníků poskládejte strom a ze stromů poskládejte les.

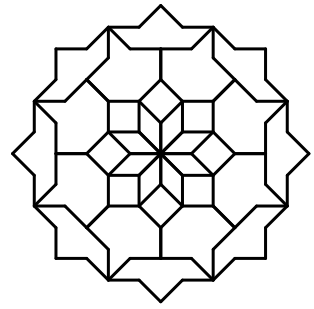


Mnohoúhelníky kolem mnohoúhelníku ⋆

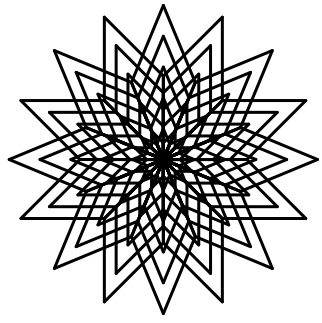
Vykreslete mnohoúhelník a kolem všech jeho hran další mnohoúhelníky.

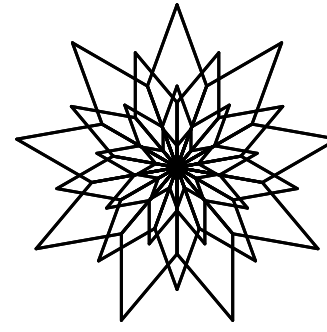


Osmičetný diamant ⋆⋆
Najděte pravidelnost v následujícím obrázku a vykreslete jej co nejkratším programem.

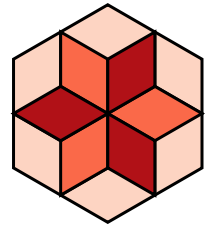


Kosočtvercové hvězdy ⋆⋆
Opakovaně z jednoho místa s různým natočením vykreslujte kosočtverce různých velikostí. Příklady pro inspiraci:





Vybarvený diamant ⋆⋆
Vykreslete jeden z „diamantů“ i s vyplňováním.



5 Želva si pamatuje proměnné

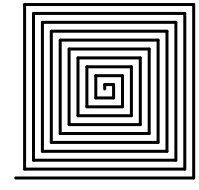
Chtěli jsme s Žofkou zkusit experiment. Již dříve jsme zvládli vykreslování mnohoúhelníků a hvězd. Ty jsme získali tak, že želva opakovaně prováděla krok dopředu a zatáčení, přičemž délka kroku i velikost úhlu byly pořád stejné. Teď nás zajímalo, co se stane, když délku kroku nebo velikost úhlu budeme měnit.

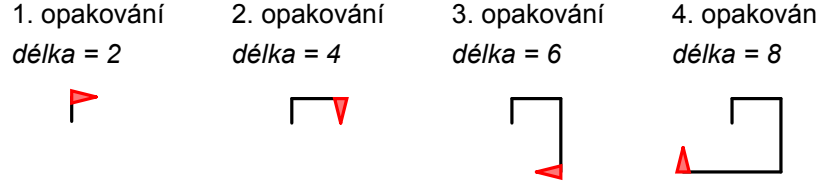
Abychom to mohli vyzkoušet, potřeboval jsem Žofce říct, aby v průběhu opakování měnila délku posunu či velikost úhlů. K tomu jsem ji musel naučit používat *proměnné*. V těch si želva udržuje hodnoty, které průběžně mění.

Spirály: změna délky kroku

Použití proměnných jsme natrénovali na drobné úpravě programu pro vykreslování čtverce. Podobně jako u vykreslování čtverce želva opakovaně provádí krok dopředu a zatočení o 90 stupňů. Nově provádí víc opakování a hlavně mění délku kroku za využití proměnné *délka*. Na začátku si Žofka do proměnné *délka* přiřadí hodnotu 2. Při každém dalším opakování dostane za úkol zvýšit hodnotu o 2, takže délky přesunů dopředu budou postupně 2, 4, 6, 8… Želva tak vykreslí spirálu v podobě „zvětšujícího se čtverce“.

```
délka ← 2
opakuj 40×
    dopředu délka
    doprava 90°
    délka ← délka + 2
```

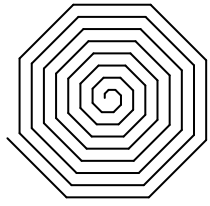




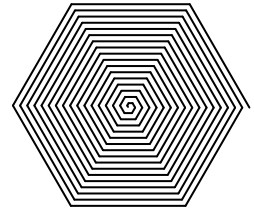
Zápis programu využívá symbol ←, abychom názorně rozlišili „přiřazení do proměnné“ od „vyjádření rovnosti“, jak se používá v matematice. Většina programovacích jazyků však pro přiřazení do proměnné používá prosté =.

Žofka měla radost, že proměnné zvládla, a chtěla s programem hned experimentovat: „Co se stane, když budu zatáčet o jiný úhel?“ Tak jsme si to vyzkoušeli. Ukázalo se, že když zvolíme úhly odpovídající mnohoúhelníkům (120°, 60°, 45°), dostaneme podobný obrázek jako v případě 90 stupňů, jen tvar spirály odpovídá příslušnému mnohoúhelníku. Podobně to funguje i pro hvězdy.

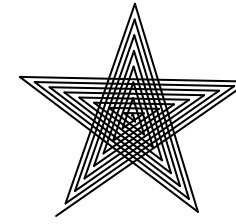
úhel 45°



úhel 60°

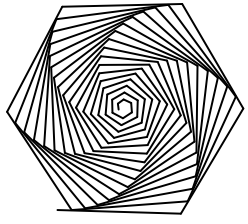


úhel 144°

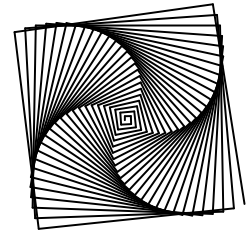


Zajímavější obrázek ale dostaneme, pokud zvolíme úhel „těsně vedle“ (např. 61°, 89°, 121°). Teď kromě spirály tvořené samotnou čárou vzniká ještě optický efekt dalších spirál tam, kde jsou čáry blízko u sebe. A i jiné úhly stojí za vyzkoušení.

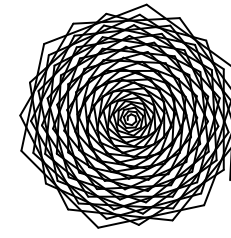
úhel 61°



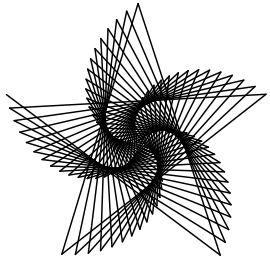
úhel 89°



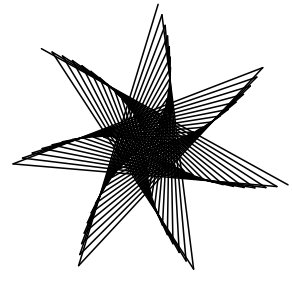
úhel 57°



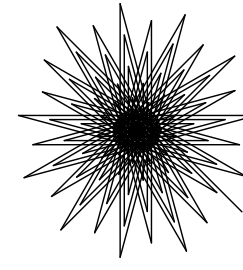
úhel 145°



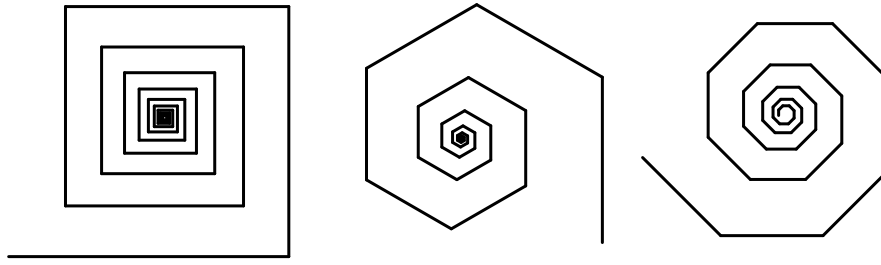
úhel 154°



úhel 165°



Doposud jsme délku zvyšovali přičítáním, můžeme ji ale zvyšovat také násobením, např. v každém kroku zvýšit délku o 10 %. To odpovídá tomu, že v programu nahradíme řádek $délka \leftarrow délka + 2$ za $délka \leftarrow 1,1 \cdot délka$. Dostáváme pak spirály jiného typu.

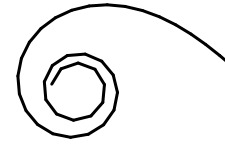


Při zvyšování proměnné přičítáním tvoří délky kroků aritmetickou posloupnost a výsledná spirála (v případě, že bychom ji udělali „kulatou“) se nazývá Archimédova spirála. Při zvyšování délek násobením tvoří délky kroků geometrickou posloupnost a odpovídající kulatá spirála se nazývá logaritmická.

Spirály: změna velikosti úhlu

Teď zkusíme zachovat délku kroku a měnit úhel. Začneme například s úhlem 40 stupňů a při každém opakování snížíme úhel o 1 stupeň. Dostaneme tak „kulatou“ spirálu, která se postupně rozpíná – vypadá podobně jako spirály, u kterých jsme zvětšovali délku násobením.

```
úhel ← 40
opakuj 40×
   dopředu 5
   doprava úhel
   úhel ← úhel - 1
```



U tohoto příkladu se po 40 opakováních dostaneme na úhel 0 stupňů, tedy želva přestane zatáčet.

„Žofko, co kdybys pokračovala v opakování ještě dál?“

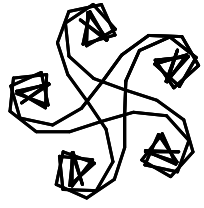
„To ale proměnná *úhel* začne být záporná. Mám zatáčet doprava o záporný úhel?“

„To přece není problém. Třeba zatáčení doprava o -5 stupňů je to stejné jako zatáčení doleva o 5 stupňů. Takže to stále dává smysl. Zkus to.“

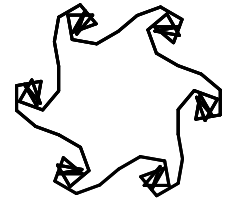
Žofka tedy začala zatáčet na druhou stranu a vytvořila ještě jednu symetrickou spirálu. Při 80 opakováních vykreslila tento obrázek:

Takto se nám tedy přičítáním konstanty k úhlu postupně mění zatáčení doprava a doleva. Můžeme zkusit i méně plynulou změnu úhlu. Pokud vhodně trefíme kombinaci počátečního úhlu a změny, dostaneme různé zajímavé variace na spirály.

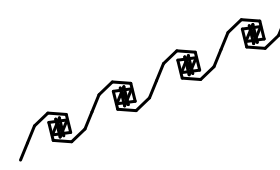
iniciální úhel 40°
změna 30°



iniciální úhel 40°
změna 30°



iniciální úhel 0°
změna 24°

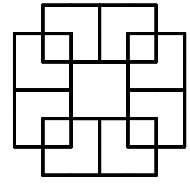


Dělitelnost

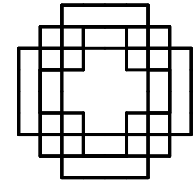
Když jsem Žofku naučil počítat, rozhodl jsem se procvičit s ní dělitelnost. Začal jsem tím nejjednodušším, co mě napadlo: Nechal jsem Žofku chodit rovně a při každém pátém kroku zatočit doleva. To ale moc zajímavé nebylo – Žofka vykreslila obyčejný čtvereček a chodila pořád dokola. Tak jsem přidal ještě jednoho dělitele. Co se stane, když bude Žofka zatáčet při každém pátém kroku doleva a při každém sedmém kroku doprava? To už je zajímavější. Podle toho, jaké zvolíme dělitele, dostáváme různé obrazce, některé celkem zajímavé:

```
k ← 1
opakuj 1000×
   dopředu 10
   pokud a dělí k
      doleva 90°
   pokud b dělí k
      doprava 90°
   k ← k+1
```

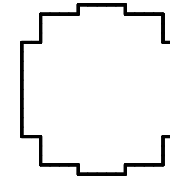
a = 2, b = 9



a = 4, b = 11

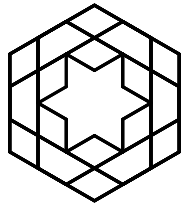


a = 5, b = 7

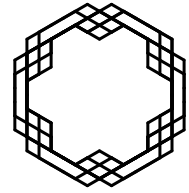


„Zatáčení o 90° už mě nebaví. Co zkusit nějaký jiný úhel?“ povídá Žofka. Tak jsme začali experimentovat s úhly. Pokud změníme úhel z 90° na 60° a zachováme dělitele, dostaneme obrazce, které mají podobné charakteristiky, ale podobnost není nijak přímočará:

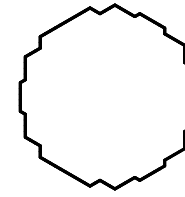
a = 2, b = 9



a = 4, b = 11



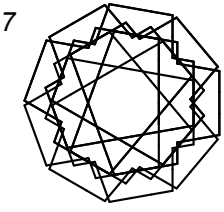
a = 5, b = 7



Pro další volby úhlů a dělitelů můžeme dostat docela komplikované obrazce – zvlášť když vezmeme v potaz, jak jednoduchým programem tyto obrazce vznikají.

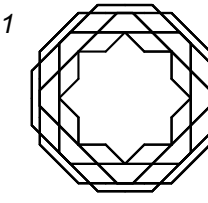
a = 3, b = 7

80°



a = 2, b = 11

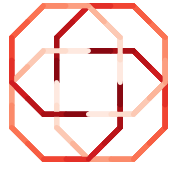
45°

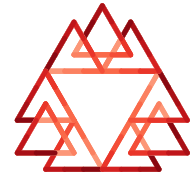


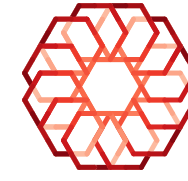
„Proč vlastně mám za úkol provést to opakování tisíckrát? To je hrozně moc práce a stejně po čase chodím dokola po stejných čarách,“ stěžovala si Žofka.

Musel jsem přiznat, že číslo 1000 nemá žádný silný důvod. Prostě jsem chtěl, aby Žofka opakovala příkazy tak dlouho, dokud se obrazec neuzavře. U základních hvězd jsem zvládl vypočítat potřebný počet opakování pomocí největšího společného dělitele. Tady bych potřeboval vypočítat počet opakování na základě úhlu a dělitelů, což ale vůbec není jednoduché. Pro některé kombinace se obrazec dokonce ani neuzavře a program začne vytvářet nekonečný vzor.

„A co to ještě trochu obarvit?“ navrhla Žofka. Tak jsme využili dělitelnost ještě na přidání odstínů červené.





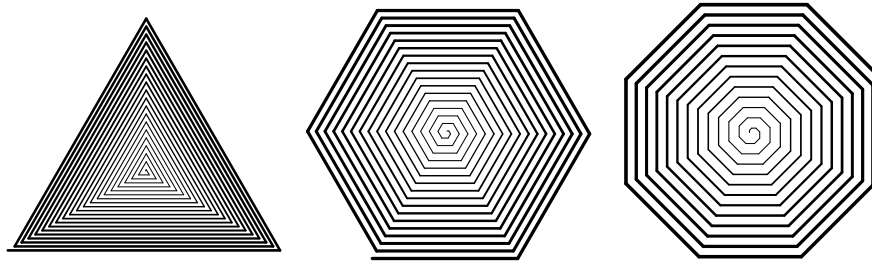


Výzvy

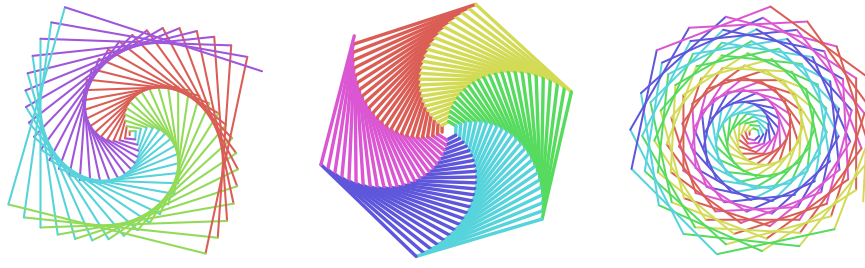
Proměnné na Umíme informatiku ⋆
Vyřešit úlohy ze sady Proměnné v systému Umíme informatiku (v blokovém programování nebo v Pythonu):
`https://www.umimeinformatiku.cz/zelvi-grafika`
`https://www.umimeinformatiku.cz/python-zelva`

Experimenty se spirálami ⋆
Pohrajte si s příklady uvedenými v textu – zkoušejte různé kombinace úhlů, délek a jejich změn v průběhu vykreslování spirály.

Spirály s proměnlivou šířkou ⋆⋆
Upravte programy uvedené v textu tak, aby tloušťka čáry postupně rostla.

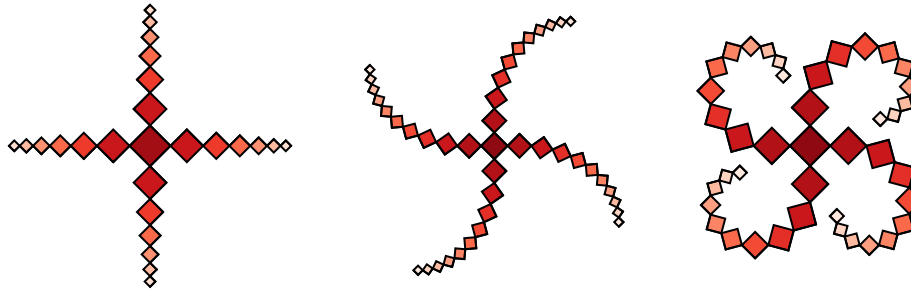


Barevné spirály ⋆⋆
Přidejte do programů uvedených v textu změnu barvy pera.



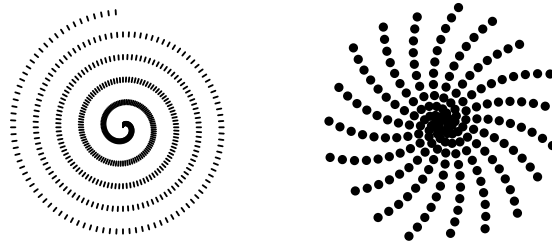
Čtvercové spirály ⋆⋆

Vykreslete řady zmenšujících se čtverců, přidejte mírné otáčení a dostanete zajímavou variaci na spirály.



Čárkované a tečkované spirály ⋆⋆

Zkuste vykreslovat spirály za využití přerušovaných čar nebo teček.



Experimenty s dělitelností ⋆⋆

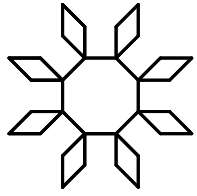
Experimentujte s programem pro klikatou čáru podle dělitelnosti. Několik námětů:

- zkoušejte různé hodnoty a, b a různé úhly,
- zkuste zatáčet v obou případech (dělitelnost a i b) doleva,
- zkuste zatáčet o různé úhly při dělitelnosti a a b,
- místo zatáčky použijte při dělitelnosti b otočku o $180°$ nebo nějakou složitější akci (např. skok kombinovaný se zatáčkou),
- přidejte dalšího dělitele c a jemu příslušnou akci (zatočení doleva, otočka, skok).

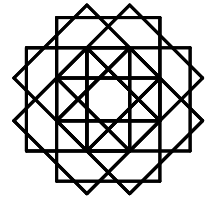
Jak vznikly obrazce? ⋆⋆

Všechny následující obrazce vznikly na základě jednoduchého programu, který využívá tři parametry (α, β, k). Odhalte, jak program funguje, a vykreslete další zajímavé obrázky pro jiné volby parametrů.

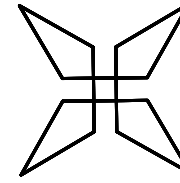
α = 45°, β = 90°, k = 5



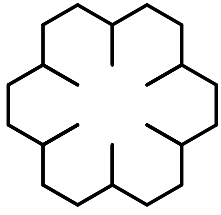
α = 90°, β = 45°, k = 5



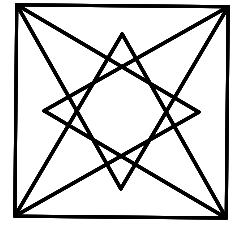
α = 60°, β = 90°, k = 3



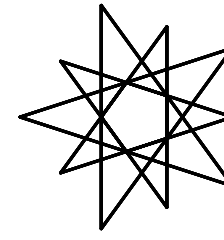
α = 60°, β = 120°, k = 5



α = 120°, β = 30°, k = 5

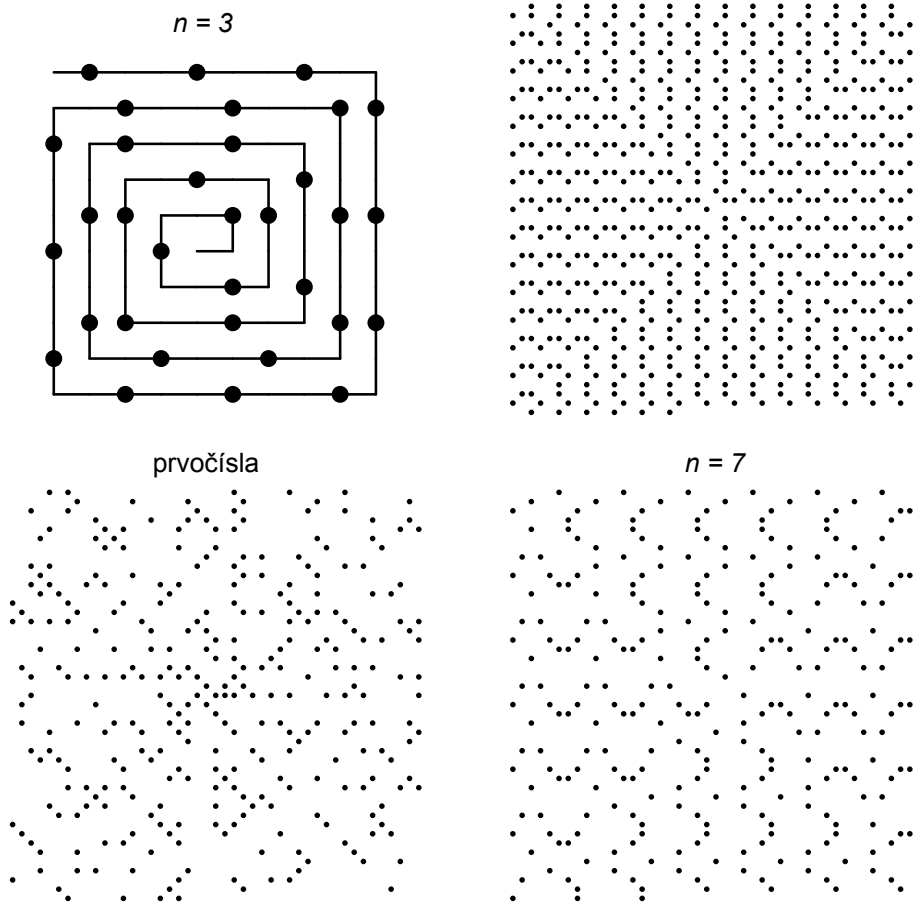


α = 144°, β = 72°, k = 4



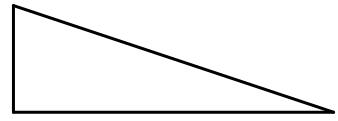
Ulamova spirála ⋆⋆⋆

Nechte želvu chodit do spirály a přitom počítat. Pokud je aktuální číslo dělitelné k, ať udělá tečku. Obrázek níže ilustruje tento postup pro $k = 3$. Pokud skryjeme čáru a necháme pouze tečky, dostaneme zajímavé vzory, jejichž podoba závisí na volbě k. Pokud místo testu dělitelnosti použijeme test prvočíselnosti, dostaneme takzvanou Ulamovu spirálu. Ta ukazuje komplikované chování prvočísel, protože obrázek nemá žádnou zcela pravidelnou strukturu, ale určité vzory se v něm najít dají, například časté diagonály.



6 Želví trénink geometrie

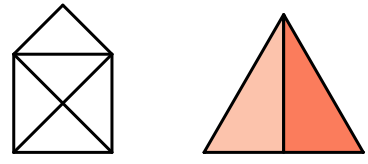
Žofka brzy hbitě zvládla opakování příkazů, použití nových příkazů a naučila se pamatovat si proměnné. Oba dva jsme byli nadšení z dosavadních pokusů, při kterých jsme pomocí jednoduchých programů zvládali vykreslit zajímavé a komplikované vzory. A tak jsme se pustili do zkoumání světa kolem nás a zkoušeli společně kreslit ledasco, co nás zaujalo. Najednou se ukázalo, že některé zdánlivě jednoduché obrázky jsou překvapivě složité. Třeba už takový obyčejný pravoúhlý trojúhelník:



Jak mají být dlouhé strany trojúhelníku a o jaký úhel má želva zatáčet? To není na první pohled vůbec jasné. Takže teď přišla s přemýšlením řada na mě. Abych dal Žofce správné pokyny, musel jsem si oprášit znalosti z matematiky.

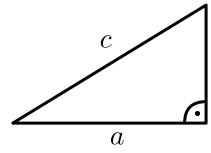
Pravoúhlé trojúhelníky

Uvažme pro začátek následující obrázky – klasický „domeček jedním tahem“ nebo rovnostranný trojúhelník rozdělený na poloviny:



U těchto obrázků je celkem jasné, o jaké úhly má želva zatáčet – u domečku to jsou vždy násobky 45 stupňů, u trojúhelníku násobky 30 stupňů. Potřebujeme však ještě vypočítat správné vzdálenosti: Jak dlouhá má být diagonála v domečku? Jak dlouhá je půlící čára v trojúhelníku? K takovým

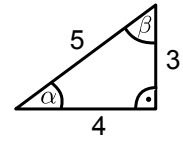
výpočtům nám poslouží Pythagorova věta, která udává vztah mezi délkami stran v pravoúhlém trojúhelníku:



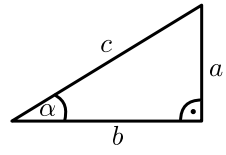
$$a^2 + b^2 = c^2$$

Pokud má tedy strana domečku délku x, diagonála musí mít délku $\sqrt{x^2 + x^2} = \sqrt{2}x$. Pokud má rovnostranný trojúhelník délku strany x, výška musí mít délku $\sqrt{x^2 - (x/2)^2} = \frac{\sqrt{3}}{2}x$.

Pokračujme s trojúhelníky dál. Řekněme, že chceme vykreslit pravoúhlý trojúhelník s odvěsnami délky 3 a 4. Pomocí Pythagorovy věty vypočítáme, že přepona má délku $\sqrt{3^2 + 4^2} = 5$. Ale to nám ještě pro nakreslení nestačí. Musíme ještě zjistit, o jaké úhly má želva zatáčet. Jaké jsou úhly α, β v tomto trojúhelníku?



Krom Pythagorovy věty musíme ještě umět využívat goniometrické funkce a cyklometrické funkce. Goniometrické funkce (sin, cos, tan) umožňují určit délku přesunu na základě znalosti úhlu. Cyklometrické funkce (arcsin, arccos, arctan) naopak umožňují určit úhel na základě délek stran. Definice funkcí znázorňuje obrázek:



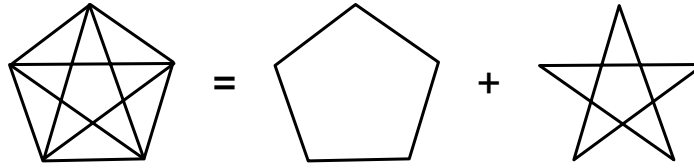
$$\sin(\alpha) = \frac{a}{c} \qquad \alpha = \arcsin(\tfrac{a}{c})$$
$$\cos(\alpha) = \frac{b}{c} \qquad \alpha = \arccos(\tfrac{b}{c})$$
$$\tan(\alpha) = \frac{a}{b} \qquad \alpha = \arctan(\tfrac{a}{b})$$

Cyklometrické funkce jsou inverzní funkce ke goniometrickým funkcím, proto se někdy zapisují jako $\sin^{-1}, \cos^{-1}, \tan^{-1}$.

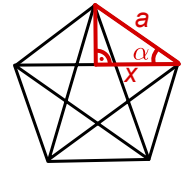
Jak nám tyto funkce pomohou? Například když chceme vypočítat úhly ve zmíněném trojúhelníku se stranami 3, 4, 5, můžeme je vypočítat jako:

$$\begin{aligned} \alpha &= \arctan(\tfrac{3}{4}) \approx 36,9^\circ \\ \beta &= \arctan(\tfrac{4}{3}) \approx 53,1^\circ \end{aligned}$$

„Tolik vzorečků a je z toho obyčejný trojúhelník," stěžovala si netrpělivě Žofka. „Zvládnout trojúhelníky je základ. Jak zvládneme trojúhelníky, hned nám půjdou dobře i složitější obrazce," odvětil jsem. A jako další krok jsme zkusili pentagram. Ten se skládá z pětiúhelníku a „hvězdičky" s pěti vrcholy:

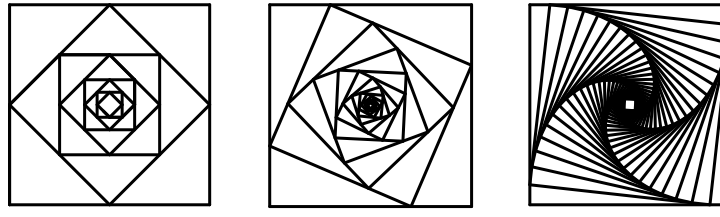


Oboje už umíme vykreslit z dřívějška, nyní je ale potřebujeme „napasovat" na sebe. Začneme tím, že vykreslíme pětiúhelník o straně a. Následně potřebujeme dovnitř zakreslit hvězdu. Musíme zjistit, o jaký úhel se máme před započetím kreslení hvězdy otočit a jakou délku ramena má hvězda mít. K tomu se hodí zvýraznit si v pentagramu následující pravoúhlý trojúhelník:



V tomto trojúhelníku potřebujeme určit α a x. Úhel α můžeme zjistit třeba následující úvahou. Vnitřní úhel pětiúhelníku je $108°$, protože vnější úhel je $360°/5 = 72°$, jak už víme z cvičení na vykreslování mnohoúhelníků. Úhel α se vyskytuje v rovnoramenném trojúhelníku, kde $108°$ je u vrcholu, takže z dopočítání do $180°$ dostáváme $\alpha = (180° - 108°)/2 = 36°$. Délku x pak dopočítáme za použití goniometrických funkcí: $x = a\cos(36°)$.

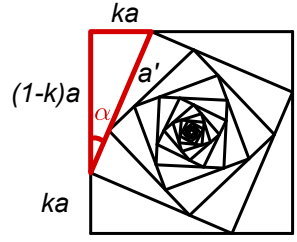
Pentagram se Žofce líbil víc než trojúhelník, ale pořád se trochu nudila. Mně trvalo dlouho, než jsem vypočítal, jaké jí mám dát příkazy, a ona pak měla obrázek nakreslený hned. Tak jsem vymyslel příklad, u kterého si zakreslila trochu víc – vnořené čtverce:



Základní způsob vykreslování je celkem jasný. Želva musí opakovaně provádět následující kroky: Vykreslit čtverec o délce a, posunout se dopředu o část a, zatočit doprava, přepočítat velikost a. Trochu těžší jsou detaily. O ja-

kou část a se má želva posunout? O jaký úhel má zatočit? Jak má správně přepočítat velikost a, tj. kolikrát je vnořený čtverec menší než ten hlavní?

Pro první verzi (obrázek vlevo) je to ještě celkem lehké. Posun je do poloviny a, zatáčka o $45°$ a aktualizaci délky vypočítáme z Pythagorovy věty jako $\sqrt{(a/2)^2 + (a/2)^2} = a/\sqrt{2}$. Pro obecný příklad nastavíme koeficient k určující část strany, o kterou se želva posune, než začne vykreslovat zmenšený čtverec (základní verze odpovídá volbě $k = 0,5$). Následně si v obrázku zvýrazníme pravoúhlý trojúhelník a z něj vypočítáme potřebný úhel α a aktualizovanou vzdálenost a'.



$$\alpha = \arctan(k/(1-k))$$

$$a' = \sqrt{(k \cdot a)^2 + ((1-k) \cdot a)^2}$$
$$= a\sqrt{1 - 2k + 2k^2}$$

Kružnice

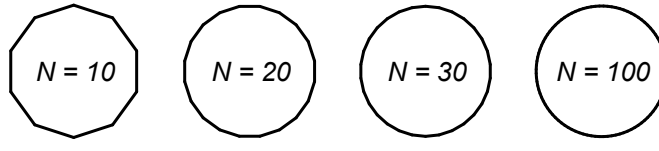
Žofka jednoho dne povídá: „Pořád kreslíme jenom obrázky z rovných čar. Já chci kreslit i něco kulatého: míč, kytky, slunce. Nauč mě příkaz pro kreslení koleček."

„To bych sice mohl, ale není to potřeba. Myslím, že zvládneš udělat kolečko i s příkazy, které už umíš," odpověděl jsem. „Stačí, když prostě uděláš hodně krátkých rovných čar. Třeba kružnici můžeš vykreslit jako mnohoúhelník s mnoha vrcholy – třeba s tři sta šedesáti."

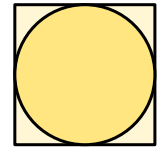
„Ale to je podvod! To přece nebude opravdová kružnice," namítla Žofka.

„A vadí to? Pro oko je to nerozpoznatelné. Třeba v knížce se taky každý obrázek skládá z mnoha malých bodů, takže bys mohla říct, že i obrázky v knížce jsou podvod. Přitom to nikomu nevadí. Pojď to zkusit a uvidíš, že to bude vypadat pěkně."

Tak jsme to zkusili a Žofka mi dala za pravdu. Uznala, že už třicetiúhelník vypadá hodně podobně jako kružnice a stoúhelník už je od opravdové kružnice k nerozeznání:



Vykreslit kulatě vypadající křivku je tedy snadné. Trochu náročnější úkol je vykreslit zakulacenou křivku, když máme přesné požadavky na její vzhled. Například když chceme vykreslit následující obrázek, potřebujeme umět vykreslit kružnici o poloměru, který je roven polovině délky strany čtverce:



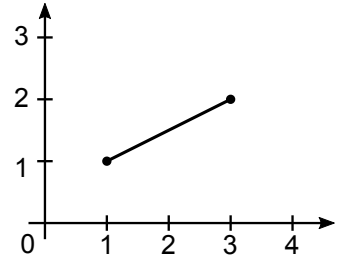
Potřebujeme tedy vyřešit následující úkol. Chceme, aby kružnice měla poloměr r, přičemž ji aproximujeme pomocí N-úhelníku. Jaká má být délka a jednotlivých kroků, které bude želva dělat, v závislosti na r a N?

K tomu můžeme dospět třeba touto úvahou: Obvod kružnice je dán vztahem $2\pi r$, my kružnici aproximujeme N-úhelníkem se stranou a, který by měl mít přibližně stejný obvod, takže délka jedné strany bude $a = 2\pi r/N$. Anebo můžeme zkusit jinou úvahu: Vypočítáme délku strany a tak, aby kružnice opsaná výslednému N-úhelníku měla poloměr r. To vede po troše rozmyšlení na vzoreček $a = 2r\sin(180/N)$. Každý vzoreček dává trochu jinou hodnotu, ale pro $N > 20$ to vychází prakticky nastejno.

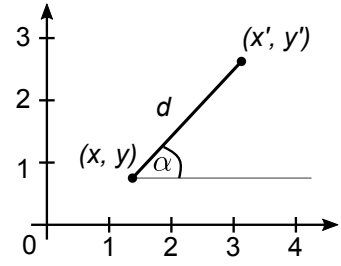
Podobně jako kreslíme kružnice, můžeme kreslit i oblouky – například oblouk velikosti $60°$ můžeme vykreslit jako šedesátiúhelník, ze kterého uděláme prvních 10 čar. Oblouky se hodí na několik z níže uvedených výzev.

Souřadnice a virtuální želva

Jednou v zimě Žofka onemocněla a na kreslení byla unavená. Tak jsem se rozhodl udělat si virtuální želvu na počítači, která měla simulovat příkazy a vykreslovat obrázky na monitor. Protože vykreslování v počítači se dělá pomocí souřadnic, musel jsem napsat program, který pro posloupnost želvích příkazů odsimuluje pohyb želvy a uloží výsledný obrázek za využití souřadnic. Například následující úsečka je uložena jako spojnice bodů $(1, 1)$ a $(3, 2)$:



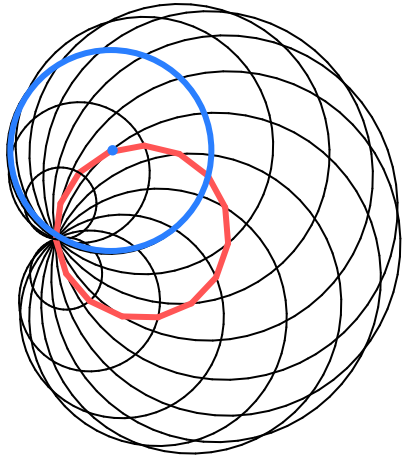
Stav želvy v každém okamžiku je dán souřadnicemi x, y a natočením α. Při provádění příkazů „doprava“ a „doleva“ se prostě jen změní úhel α. Při přesunu dopředu o d musíme vypočítat nové souřadnice x', y' a uložit do výsledného obrázku čáru z (x, y) do (x', y'). Nové souřadnice x', y' vypočítáme za využití goniometrických funkcí:

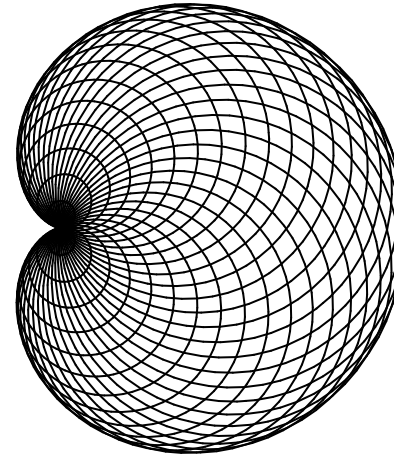


$$\begin{aligned} x' &= x + d \cdot \cos(\alpha) \\ y' &= y + d \cdot \sin(\alpha) \end{aligned}$$

Srdcovka

Na závěr naší exkurze do geometrie jsme se pustili do zajímavého obrazce, při jehož vykreslování jsme využili všechny předchozí zkušenosti. Nechal jsem Žofku putovat po N-úhelníku (na obrázku znázorněn červeně) a v každém vrcholu jsem ji nechal vykreslit kružnici se středem v aktuálním vrcholu a poloměrem odpovídajícím vzdálenosti od počátku (místa, kde začala). Na obrázku je modře zakreslena kružnice pro 3. vrchol mnohoúhelníku:





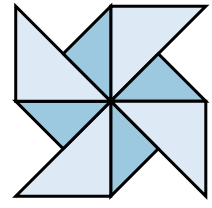
Pro velké N dostáváme čím dál vyplněnější útvar, který má zajímavou obrysovou křivku. Podle svého tvaru se nazývá „kardioida“ nebo v češtině též „srdcovka“. Objevuje se v mnoha různých souvislostech, například když po sobě kutálíme dvě kružnice a sledujeme polohu jednoho z bodů, nebo také ve známém Mandelbrotově fraktálu.

Aby se nám povedlo obrazec vykreslit, potřebovali jsme si ujasnit několik věcí. Jednak jsem musel Žofku naučit vykreslit kružnici se středem v aktuální pozici. To bylo celkem snadné: „Udělej úhyb o r, vykresli kružnici s poloměrem r, udělej úhyb o $-r$“ (úhyb jsme se naučili již dříve, viz strana 23). Dále bylo potřeba zjistit pro každou kružnici správný poloměr. Abych si to usnadnil, naučil jsem Žofku příkaz „přiřaď do proměnných x, y svoje aktuální souřadnice“. Za využití těchto souřadnic pak už bylo snadné vypočítat potřebný poloměr z Pythagorovy věty: $\sqrt{x^2 + y^2}$. Po troše přemýšlení jsem ale vymyslel i alternativní způsob, jak poloměr vypočítat – pomocí rekurzivního vzorečku, který udává délku i-té diagonály v N-úhelníku: $d(N, i) = a\cos(i \cdot 180/N) + d(N, i-1)\cos(180/N)$. Což mi připomnělo, že je nejvyšší čas zasvětit Žofku do tajů rekurze.

Výzvy

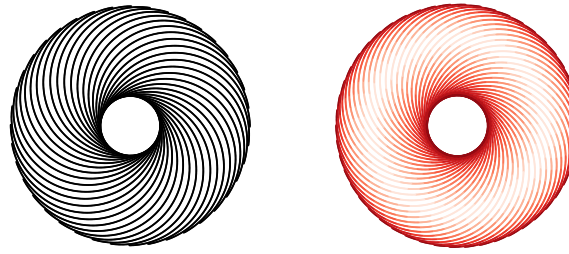
Správné úhly na Umíme informatiku ⋆
Vyřešte úlohy ze sady Správné úhly v systému Umíme informatiku (v blokovém programování nebo v Pythonu):
`https://www.umimeinformatiku.cz/zelvi-grafika`
`https://www.umimeinformatiku.cz/python-zelva`

Větrník ⋆
Vykreslete zjednodušený obrázek papírového větrníku.



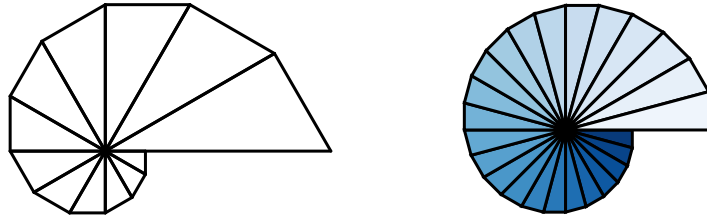
Americká kobliha ⋆⋆

Vykreslete následující obrázek, připomínající americkou koblihu (donut). Pro lepší efekt použijte stínované barvy. Nápověda: Obrázek se skládá z mnoha půlkružnic.



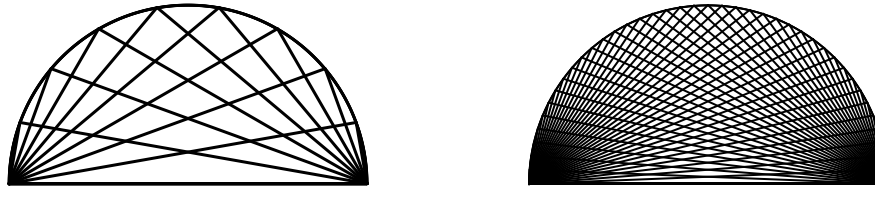
Trojúhelníková ulita ⋆⋆

Z navazujících trojúhelníků vytvořte ulitu. Pro lepší efekt můžete trojúhelníky vybarvit odstíny jedné barvy.



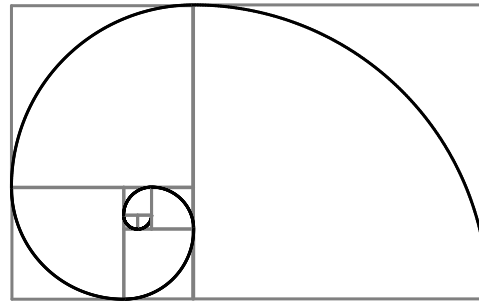
Thaletova věta ⋆⋆

Thaletova věta říká: „Všechny obvodové úhly sestrojené nad průměrem kružnice jsou pravé.“ Vytvořte obrázky ilustrující toto tvrzení.



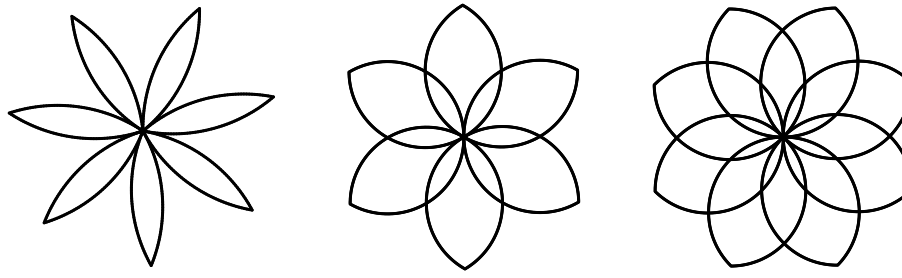
Fibonačiho spirála ⋆⋆

Fibonačiho posloupnost začíná dvěma jedničkami a každý další prvek je součet dvou předchozích: 1, 1, 2, 3, 5, 8, 13, 21... Geometricky tuto posloupnost dostaneme tak, že dáme vedle sebe dva čtverce velikosti 1 a pak vždy přikreslíme další čtverec, aby sousedil se dvěma předchozími. Pokud do těchto čtverců přidáme čtvrtkruhy, dostaneme Fibonačiho spirálu.



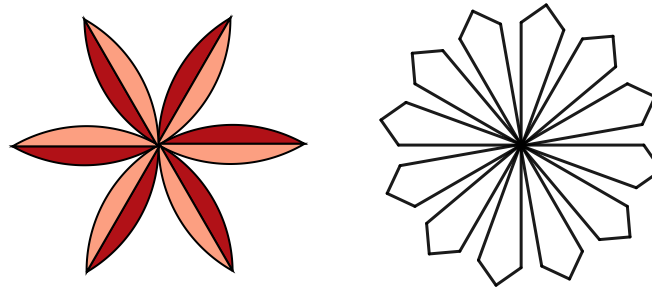
Kytičky ⋆⋆

Pomocí oblouků vykreslete kytičky. Vzhled kytičky je určen dvěma parametry: velikostí oblouku okvětních lístků a jejich početem.



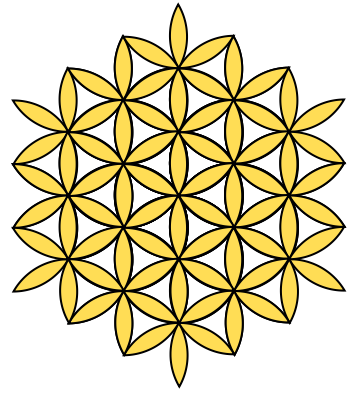
Variace na kytičky ⋆⋆

Vyzkoušejte různé variace na kytičky, např. vybarvené či hranaté. Kytičkám můžete přidat i stonek a vykreslit celou louku.



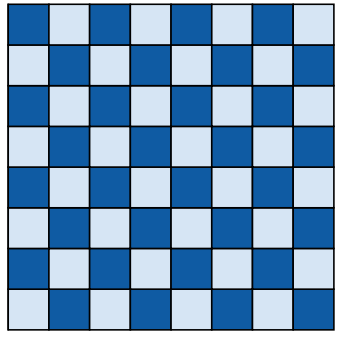
Květ života ⋆⋆⋆

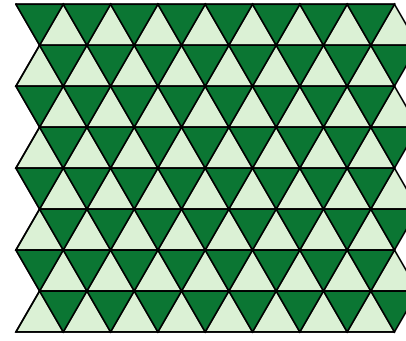
Vykreslete následující vzor, který bývá označován jako „květ života“.

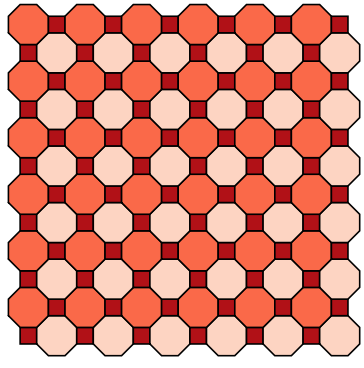


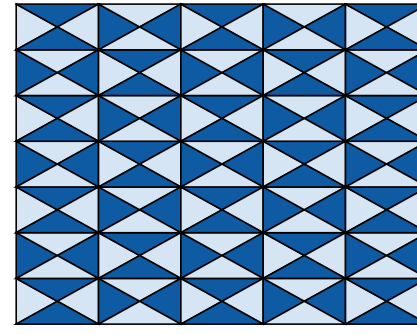
Teselace ⋆⋆ –⋆⋆⋆

Teselace („dlaždičkování“) je pokrytí roviny pomocí jednoho nebo více geometrických útvarů. Zkuste vykreslit třeba některé z následujících základních vzorů. Další náměty snadno najdete na internetu při hledání anglických pojmů „tilings“ a „tesellation“.







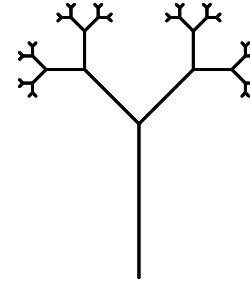


7 Želva zkoumá rekurzi a fraktály

K opravdu zajímavým obrázkům jsme se s Žofkou dostali ve chvíli, kdy jsem ji naučil rekurzi. Nebylo to snadné, protože rekurze je náročná na pochopení a vyžaduje od želvy, aby při provádění programu dávala dobrý pozor a neztratila se v něm. Ještěže je Žofka tak šikovná.

Rekurze znamená, že příkaz použijeme při jeho vlastní definici. To může znít poněkud podivně, a také si Žofka při prvním setkání s rekurzí stěžovala. Učil jsem ji nový příkaz pro vykreslování jednoduchého keře:

```
nauč se keř délka
  dopředu délka
  pokud délka > 10
    doleva 45°
    keř délka / 2
    doprava 90°
    keř délka / 2
    doleva 45°
  dopředu -délka
```



„To přece nejde – když mě učíš příkaz keř, nemůžeš přitom použít příkaz keř. Ten přece ještě neumím," namítla Žofka.

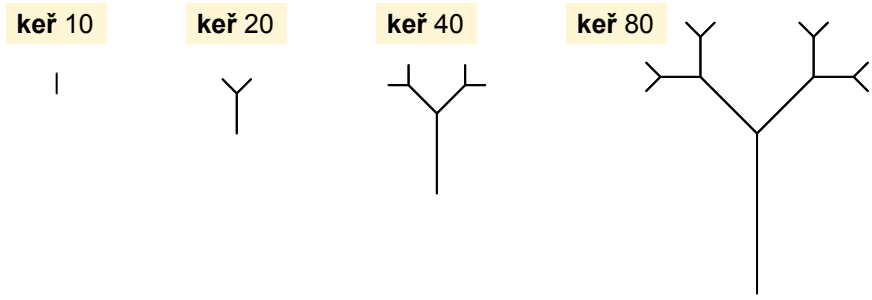
„Ale můžu. Však to zkus, prostě když narazíš na příkaz keř, použiješ definici, jako bys ji už znala," zkoušel jsem ji přesvědčit.

„Ale to se přece zacyklím a budu na věky věků kreslit keř. Takhle jsem si želví život nepředstavovala," nezdálo se to stále Žofce.

„Neboj se. Víš co, zkusíme to postupně," navrhl jsem.

A tak jsme zkusili nejprve „**keř** 10". To nebyl problém. Podmínka „pokud *délka* $>$ 10" neplatí, takže Žofka prostě jen vykreslila čáru dopředu a zase se vrátila zpět. Pak jsme zkusili „**keř** 20". Teď udělala Žofka čáru délky 20 a následně měla za úkol vykreslit dvakrát „**keř** 10" – to už ale věděla, že zvládne. Takže bez obav provedla „**keř** 10", což vedlo na vykreslení prosté

čáry, čímž celkově vykreslila malinký stromeček se dvěma větvemi. Pak jsme zkusili „**keř** 40“, což už opět při znalosti toho, co dělá „**keř** 20“, nebylo těžké.



Tento příklad je typickou ilustrací rekurze. Rekurze v kódu odpovídá rekurzivní struktuře obrázku – keř se skládá ze dvou menších kopií keře. Program tedy dělá to, že vykreslí stonek keře a pak se vhodně otočí a „rekurzivně“ zavolá vykreslení dvou menších verzí keře. Takové rekurzivní zanořování nemůže probíhat věčně, jinak by želva obrázek nikdy nedodělala. Rekurzivní příkazy tedy musí obsahovat takzvanou „koncovou podmínku“, která rekurzivní volání ukončí – v našem případě to je podmínka „pokud $délka > 10$“.

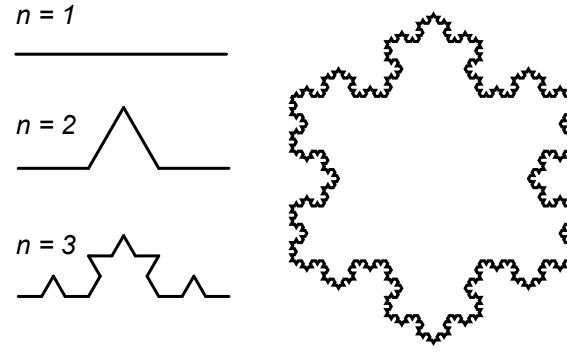
Ke správnému fungování rekurze je v případě želví grafiky typicky potřeba, aby se želva na konci rekurzivního příkazu vrátila tam, kde začala. Proto v programu pro vykreslování keře potřebujeme poslední příkaz zajišťující couvání dozadu. Tímto příkazem sice nevykreslíme do obrázku žádnou novou čáru, ale pro fungování rekurze je nezbytný. Však si vyzkoušejte, co se stane, když jej vynecháme.

Fraktály

Když jsme natrénovali rekurzi, mohli jsme se pustit do fraktálů. Fraktály jsou obrázky, které jsou „soběpodobné“ – pokud obrázek zvětšujeme, dostáváme stále stejný (nebo podobný) základní motiv. Fraktály jsou velmi úzce spojeny s rekurzí. Často můžeme velmi komplikovaně vypadající fraktální obrazce vykreslit pomocí jednoduchých rekurzivních pravidel.

Průzkum fraktálů jsme s Žofkou zahájili vykreslováním Kochovy křivky a Sierpińského trojúhelníku, což jsou jedny z nejznámějších fraktálů. Kochova křivka vznikne tak, že úsečku nahrazujeme čtyřmi úsečkami o třetinové velikosti. Ze tří křivek pak můžeme poskládat pěknou vločku.

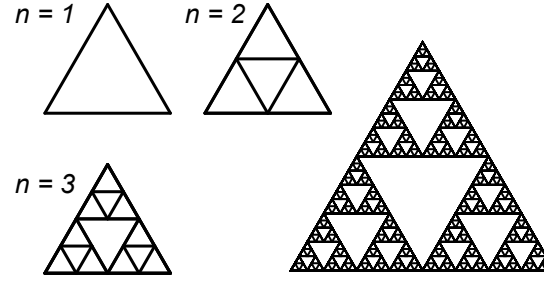
```
nauč se koch n, délka
  pokud n = 1
    dopředu délka
  jinak
    koch n-1, délka/3
    doleva 60°
    koch n-1, délka/3
    doprava 120°
    koch n-1, délka/3
    doleva 60°
    koch n-1, délka/3
```



U fraktálů typicky mluvíme o „stupni zanoření“ – jak detailně obrázek vykreslujeme. V programech pro Žofku jej označuji n. Výsledné obrázky pro různá n názorně ukazují proces vzniku fraktálů a jejich rekurzivní strukturu.

Sierpińského trojúhelník je definován tak, že vezmeme trojúhelník, z něj vyřízneme prostřední trojúhelník a tento postup opakujeme rekurzivně na zbylé tři trojúhelníky. Pro vykreslení postupujeme tak, že Sierpińského trojúhelník stupně n složíme ze tří Sierpińského trojúhelníků stupně $n - 1$ poskládaných do jednoho velkého trojúhelníka.

```
nauč se sierpiński n, délka
  pokud n = 1
    trojúhelník délka
  jinak
    opakuj 3×
      sierpiński n-1, délka/2
      dopředu délka
      doleva 120
```



Pro praktické vykreslení musíme uvažovat pouze konečný počet zanoření, a to nepříliš vysoký, protože počet čar, ze kterých se obrázky skládají, roste velmi rychle (exponenciálně). Matematicky jsou ale uvedené fraktály definovány jako výsledek nekonečného počtu zanoření. Ač se to nezdá, jsou výsledné objekty dobře definované. Mají podivuhodné vlastnosti – například Sierpińského trojúhelník je vlastně zamotaná křivka, která se dotýká sama sebe v každém bodě.

L-systémy

„Ty fraktály jsou pěkné obrázky, ale provádění rekurzivních příkazů mě unavuje," stěžovala si Žofka, „musím si pořád dávat pozor, abych to nepopletla." Moc jsem se jí nedivil, z toho zanořování a vynořování z rekurzivních volání se nejen želvě, ale i člověku snadno zamotá hlava. Abych Žofce usnadnil život, rozhodl jsem se prozkoumat L-systémy.

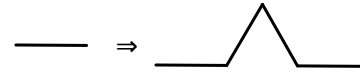
L-systémy (neboli plným názvem Lindenmayerovy systémy) byly původně navrženy pro modelování růstu rostlin, lze je ale využít i pro vykreslování celé řady zajímavých fraktálů. Jsou to takzvané „paralelní přepisovací gramatiky" založené na přepisování řetězců – používáme abecedu symbolů, máme výchozí řetězec písmen (axiom) a přepisovací pravidla, která udávají, jak se řetězec přepisuje.

„Vůbec nechápu, o čem mluvíš. Hlavně vůbec nechápu, co to má společného s obrázky a se mnou," namítla Žofka.

„Výsledný řetězec předložím tobě a ty podle něj už přímočarým způsobem vykreslíš obrázek. Pořád tam zůstane rekurze, ale ta teď bude skryta v přepisovacích pravidlech, takže ti nebude motat hlavu," zkusil jsem jí to objasnit.

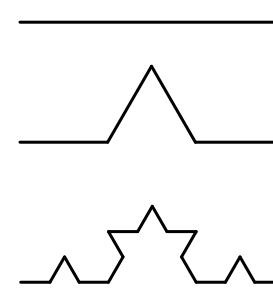
„To zní dobře, ale stejně mi pořád vůbec není jasné, co mají znamenat ty přepisovací pravidla, gramatiky, axiomy. To zní všechno děsně složitě."

„Může to znít složitě, ale základní princip je celkem jednoduchý. Ukážu ti to na příkladu Kochovy křivky."

L-systém	Význam
symboly: F	posun dopředu
-	zatočení doprava o 60°
+	zatočení doleva o 60°
axiom: F	
přepisovací pravidlo: F ⇒ F+F--F+F	

Začneme s axiomem (F). Na ten aplikujeme přepisovací pravidlo. To funguje tak, že řetězec na levé straně pravidla (F) nahradíme za řetězec na pravé straně pravidla (F+F–F+F). Tím dostaneme nový řetězec a na něj opět aplikujeme přepisovací pravidlo. Důležité je, že pravidlo použijeme vždy „paralelně", tj. všechny výskyty symbolu F přepíšeme současně. Po zadaném počtu opakování skončíme. Výsledný řetězec pak želva vykreslí podle zadaného významu symbolů.

1. iterace	F
2. iterace	F+F--F+F
3. iterace	F+F--F+F+F+F--F+F--F +F--F+F+F+F--F+F



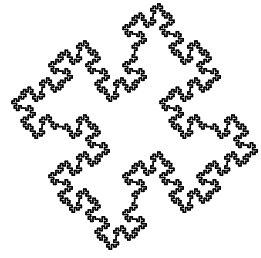
Pomocí L-systémů můžeme snadno vykreslit třeba variace na známé fraktály:

Kochův ostrov

axiom: F-F-F-F

pravidlo:
F ⇒ F-F+F+F
F-F-F+F

úhel: 90°

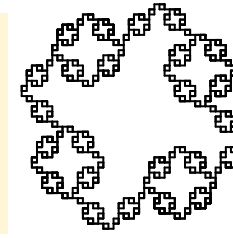


Kochovy čtverce

axiom: F-F-F-F

pravidlo:
F ⇒ F-F+F+F
F-F-F+F

úhel: 90°

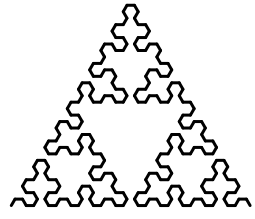


Sierpieńského fraktál

axiom: A

pravidla:
A ⇒ B-A-B
B ⇒ A+B+A

úhel: 60°

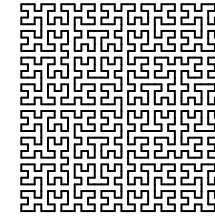


Hilbertova křivka

axiom: A

pravidla:
A => +BF-AFA-FB+
B => -AF+BFB+FA-

úhel: 90

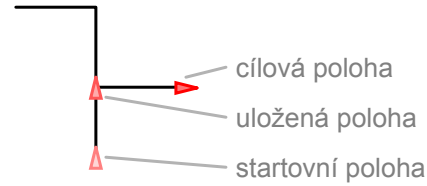


Generování rostlin

L-systémy se využívají především pro vytváření obrázků rostlin – ty mají často díky svému větvení fraktální strukturu. Než jsme se ale mohli pustit do generování obrázků rostlin, musel jsem ještě Žofku naučit nové příkazy pro ukládání a obnovování polohy:

- příkaz „ulož polohu“: želva si uloží aktuální polohu, tj. aktuální souřadnice a natočení,
- příkaz „obnov polohu“: želva obnoví poslední uloženou polohu, přičemž na příslušné místo se vrátí „skokem“, tj. bez vykreslení čáry.

```
dopředu(50)
ulož polohu
dopředu(50)
doleva(90)
dopředu(50)
obnov polohu
doprava(90)
dopředu(50)
```

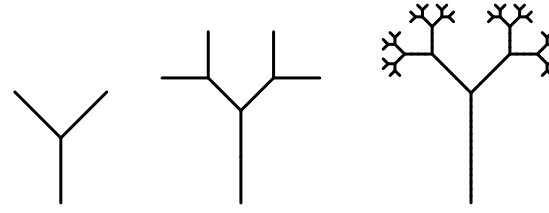


Želva si může uložit i více poloh za sebou, při obnovování se pak polohy berou od poslední (informatici používají pro označení datové struktury, do které se tato informace ukládá, termín „zásobník“). V L-systému ukládání polohy značíme symbolem `[` a obnovování polohy značíme symbolem `]`. S tímto rozšířením jsme se již mohli pustit do generování rostlin. Jako rozcvičku jsme udělali keř, na kterém jsme původně trénovali rekurzi:

úhel: 45°

A ⇒ F[+A]-A

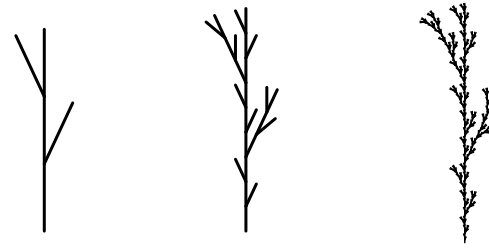
F ⇒ FF



Pak jsme zkusili trochu složitější pravidla a vyšla nám celkem reálně vypadající rostlina:

úhel: 25°

F ⇒ FF+[+F-FF]
-[-F+F+F]

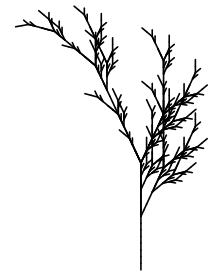


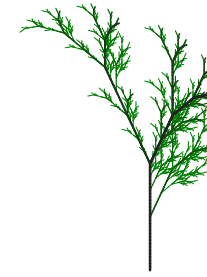
„To je pěkné, ale málo zelené. Rostliny mají být zelené,“ podotkla Žofka. Tak jsme to ještě trochu vylepšili. Přidali jsme do vykreslování proměnlivou tloušťku čáry a odstín barvy. Nastavení tloušťky a barvy dostala Žofka za úkol udělat podle hloubky „zanoření“ (což je počet uložených poloh na zásobníku).

úhel: 25°

A ⇒ F-[[A]+A]+
F[+FA]-A

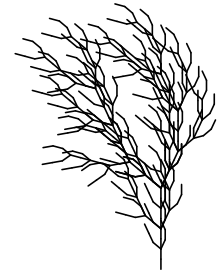
F ⇒ FF





úhel: 25°

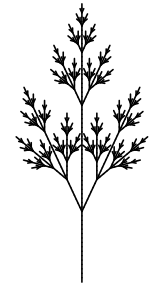
F ⇒ FF+[+F-FF]
-[-F+F+F]

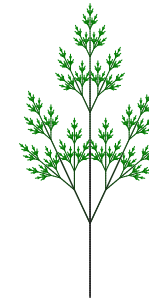


úhel: 25°

A ⇒ F[+A][-A]F[A]

F ⇒ FF

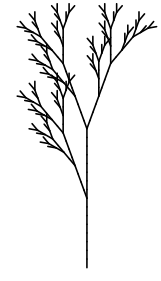


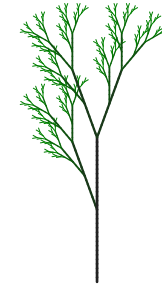


úhel: 20°

A ⇒ F[+A]F[-A]+[A]

F ⇒ FF

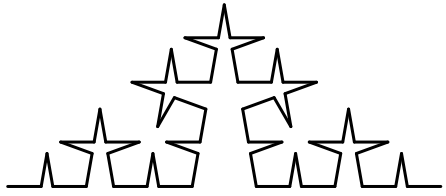


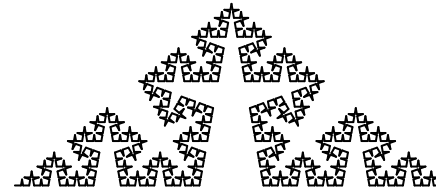


Výzvy

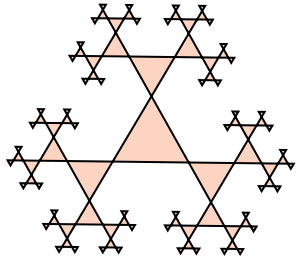
Rekurze na Umíme informatiku ⋆⋆
Vyřešte úlohy ze sady Rekurze a fraktály v systému Umíme informatiku (v Pythonu):
`https://www.umimeinformatiku.cz/python-zelva`
Pomocí blokového programování si můžete zkusit Želví experimentárium, ve kterém jsou připravené hotové programy, se kterými můžete dělat pokusy (např. měnit hodnoty parametrů):
`https://www.umimeinformatiku.cz/zelvi-grafika`

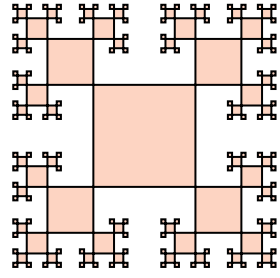
Kochovy variace ⋆⋆
Upravte vykreslování Kochovy křivky, aby místo zatáčení o 60 stupňů bylo možné použít i jiné úhly.

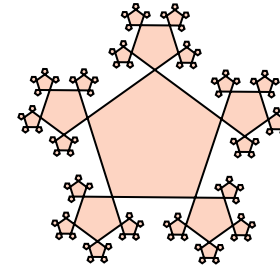




Mnohoúhelníky s fraktální ozdobou ⋆⋆⋆
Vykreslete obrázky následujícího typu: Uděláme mnohoúhelník, kolem každého vrcholu zmenšené mnohoúhelníky, stejně postupujeme dále, pěkně rekurzivně:

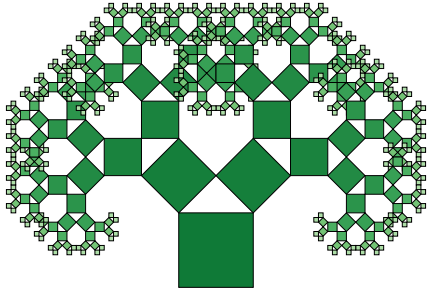


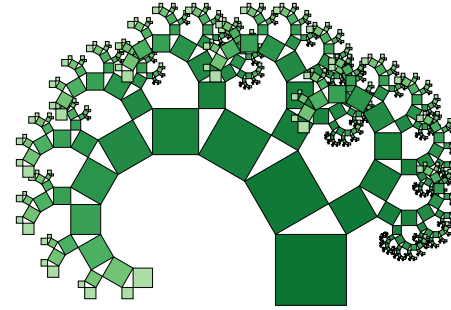




Pythagorův strom ⋆⋆⋆

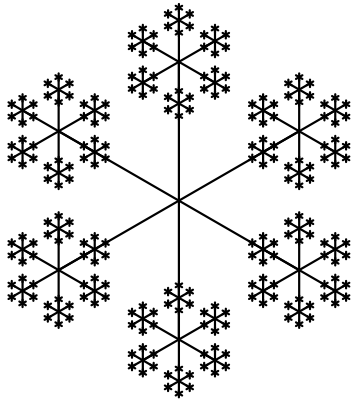
Vykreslíme čtverec, nad ním pravoúhlý trojúhelník, nad odvěsnami další čtverce, nad nimi opět pravoúhlé trojúhelníky a tak dále. Fraktál se nazývá Pythagorův strom, protože je tvořen čtverci nad trojúhelníky, podobně jak se kreslí při ukázkách Pythagorovy věty. Podle toho, jaký zvolíme tvar pravoúhlého trojúhelníku, dostáváme různě nakloněné stromy.

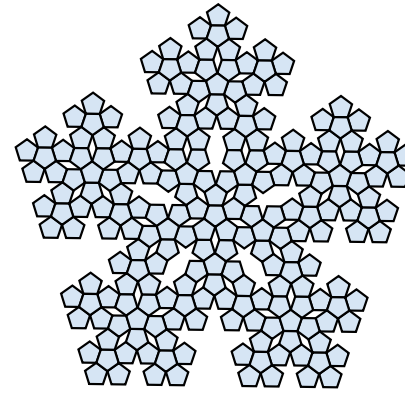




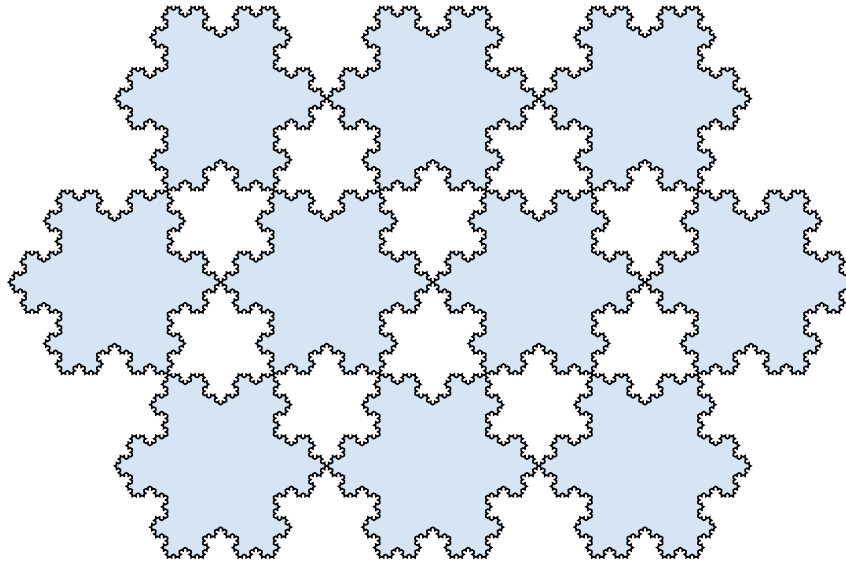
Fraktální vločky ⋆⋆⋆

Vykreslete fraktální vločky podle následujících obrázků. Nápovědy: Čárkovaná vločka je jednodušší a lze ji snadno upravit i pro jiný počet větvení než 6. Na vykreslení pětiúhelníkové vločky se hodí trocha počítání s goniometrickými funkcemi.





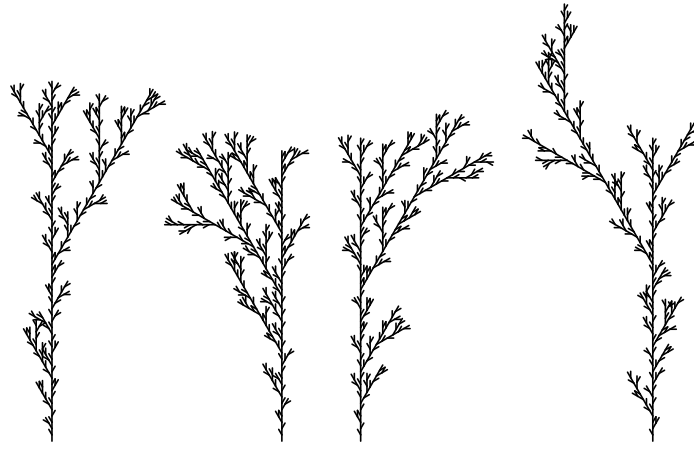
Kochovy dlaždice ⋆⋆⋆
Když poskládáme vedle sebe Kochovy vločky, vznikne zajímavé pokrytí roviny, které je tvořené z velkých a malých vloček.



L-systémy s náhodou ⋆⋆⋆
Abychom získali pomocí L-systému ještě reálněji vypadající rostliny, hodí se přidat do generování trochu náhody. K tomu slouží „stochastické L-systémy“, které mohou mít pro přepisování jednotlivých písmen uvedeno více pravidel. Při přepisování se pak vždy náhodně vybere jedno z pravidel. Pro inspiraci uvádíme příklad jednoduchých pravidel a několik výsledných roslinek, které pomocí nich můžeme dostat.

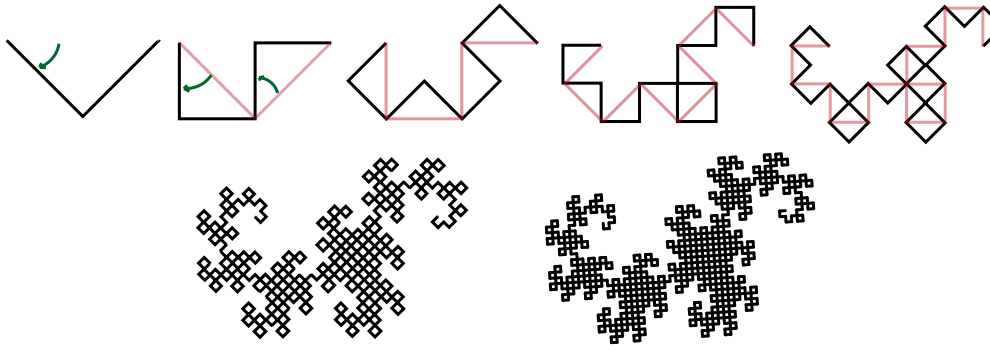
úhel: 30°

F ⇒ F[+F]F[-F]F
F ⇒ F[+F]F
F ⇒ F[-F]F



Dračí křivka ⋆⋆⋆

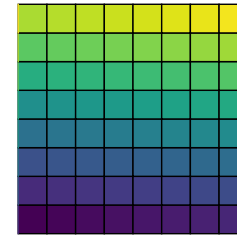
Dračí křivka je další z velmi známých fraktálů a má mnoho zajímavých vlastností. Na obrázku je zachyceno několik prvních iterací vytváření dračí křivky. Podle tohoto obrázku zkuste odvodit pravidla pro L-systém a vykreslit detailní obrázek.



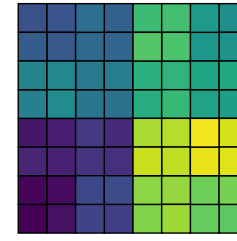
Rekurzivní dlaždice ⋆⋆⋆

I s obyčejnými čtvercovými dlaždicemi můžeme dostat zajímavé obrázky. Základní způsob, jak vykreslit čtvercové dlaždice, je pomocí dvou vnořených opakování. Vykreslování ale můžeme zapsat i rekurzivně – výsledek pak vypadá úplně stejně, jen se liší pořadí, v jakém želva čtverečky vykresluje. Na obrázku je toto pořadí znázorněno barvou.

```
opakuj N×
    opakuj N×
        čtverec délka
        úhyb délka
    úhyb -N*délka
    dopředu délka
```

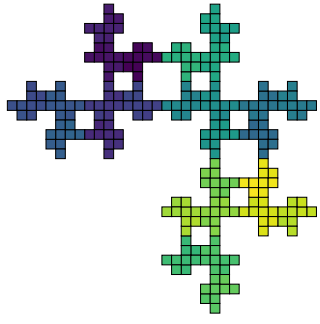


```
definuj mřížka N, délka
    pokud n = 1
        čtverec délka
    jinak
        opakuj 4×
            mřížka N-1, délka/2
            dopředu délka
            doprava 90°
```

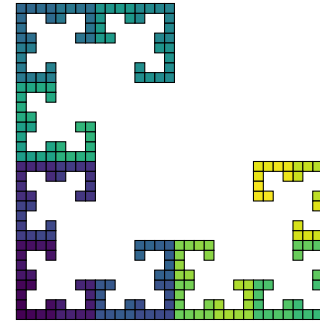


S rekurzivní verzí však můžeme daleko lépe experimentovat. Stačí třeba při opakování jedno z rekurzivních volání vynechat a dostaneme zajímavé obrazce. Zkuste tyto obrazce vytvořit a vykreslit další podobné variace.

vynecháváme 1. volání

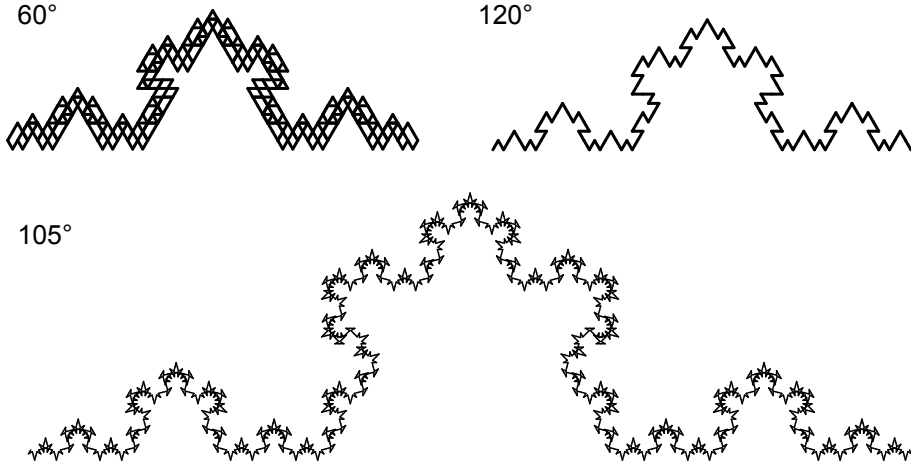


vynecháváme 3. volání



Thue-Morse a Koch ⋆⋆⋆

Thue-Morseova posloupnost je posloupnost nul a jedniček definovaná následovně: Začneme nulou a pak opakovaně připojujeme „komplement“ dosavadní posloupnosti (prohodíme nuly za jedničky a naopak). Začátek vytváření posloupnosti tedy vypadá následovně:
$0 \to 01 \to 0110 \to 01101001 \to 0110100110010110$
Posloupnost je definovaná rekurzivně a vyskytují se v ní zjevné rysy soběpodobnosti, takže je očekávatelné, že by mohla mít souvislost s fraktály. Co už není na první pohled patrné, je hluboká souvislost této posloupnosti s Kochovou křivkou. Když výslednou posloupnost interpretujeme želví grafikou, dostáváme rozličné obrázky, které však typicky silně připomínají Kochovu křivku. V následujících ukázkách je nula interpretována jako **dopředu** a jednička jako **doprava** (o uvedený úhel). Vyzkoušejte další způsoby, jak pomocí želví grafiky vykreslit Thue-Morseovu posloupnost.

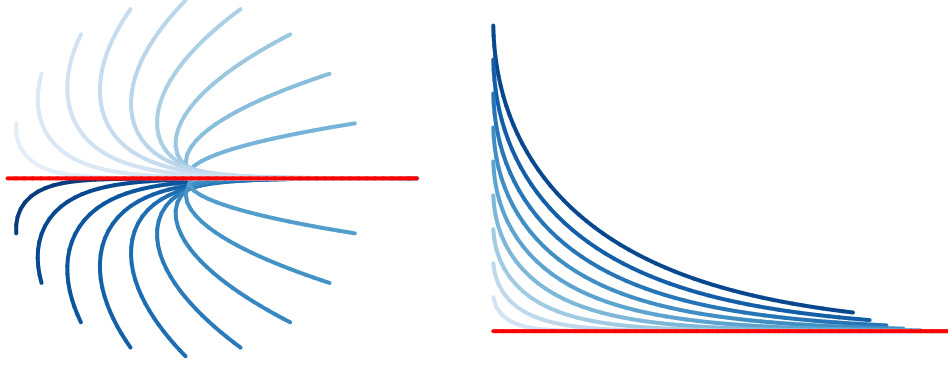


8 Želva má kamarádky

Pořídil jsem Žofce kamarádky. Žofka je rychle naučila vše, co už uměla. Potom jsme vymysleli nové příkazy, které využívaly přítomnosti více želv. To nám umožnilo vykreslovat nové zajímavé vzory, které by pro samotnou Žofku byly příliš komplikované.

Honička

Začali jsme s příkazem „otoč se směrem k jiné želvě“. Tento příkaz jsme používali hlavně na různé honičky. Rozestavil jsem třeba želvy do kruhu nebo na čáru a Žofce jsem dal za úkol posunovat se pomalu přímo rovně. Ostatní želvy ji pronásledovaly – tedy opakovaně prováděly: „Otoč se směrem k Žofce“ a „udělej malý krok“. Přitom vykreslily docela zajímavé vzory (Žofka dělá červenou čáru, ostatní želvy používají odstíny modré):



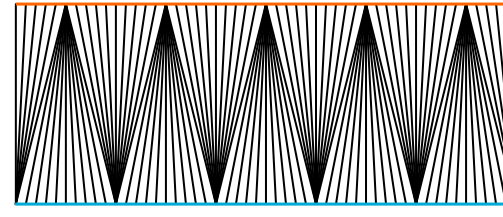
„Proč všechny želvy honí mě? Já chci taky někoho honit,“ domáhala se Žofka. Tak jsme zkusili cyklickou honičku. Želvy jsem očísloval a dal jim tento úkol: Želva číslo 1 honí želvu číslo 2, želva číslo 2 honí želvu číslo 3, želva číslo 3 honí želvu číslo 4 a tak dále, až poslední želva honí želvu číslo 1.

Když jsem želvy rozmístil do kruhu a spustil cyklickou honičku, vykreslily pěknou sérii spirál. Ještě hezčí obrázek jsme dostali, když se želvy pohybovaly různou rychlostí.

Spojnice

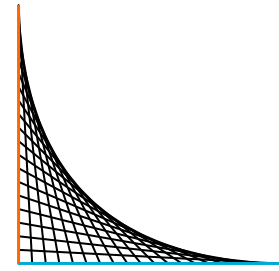
Další zajímavý příkaz je „vykresli spojnici dvou želv“. Tento příkaz měl na starosti trpaslík Tonda, který vždy seskočil z krunýře, narýsoval spojnici mezi želvami a pak si zase nasedl. Za využití tohoto příkazu jsme pak vykreslili řadu obrázků. Jako rozcvičku jsem postavil dvě želvy kousek od sebe a nechal je chodit rovně dopředu, jenom „přerušovaně“: Jedna želva udělá několik kroků, pak se zastaví a druhá se posune dopředu, pak se posune zase první. Takto pořád dál, přičemž po každém kroku vykreslíme spojnici. Dostaneme z toho zajímavý optický efekt:

```
úhyb 100
opakuj 5×
   opakuj 5×
      dopředu 10
      vykresli spojnici želv
   opakuj 10×
      dopředu 10
      vykresli spojnici želv
   opakuj 5×
      dopředu 10
      vykresli spojnici želv
```



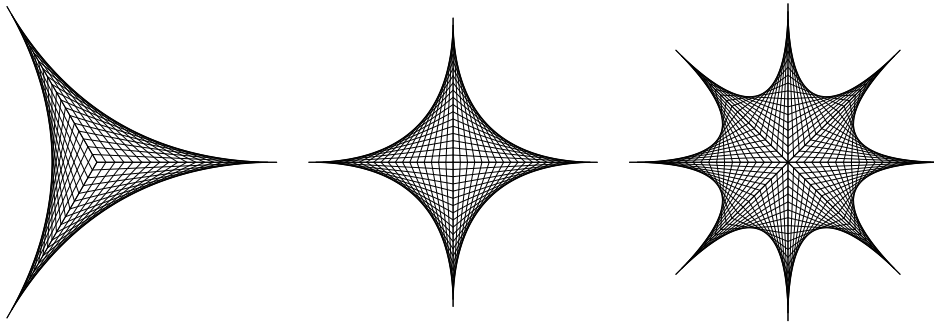
Nebo jsem nechal dvě želvy jít kolmo k sobě, přičemž tentokrát se pohybovaly současně, tj. vždy udělaly obě malý krok a pak jsme vykreslili spojnici:

```
doprava 90°
dopředu 150
doprava 180°
opakuj 15×
   vykresli spojnici želv
   dopředu 10
   dopředu 10
```

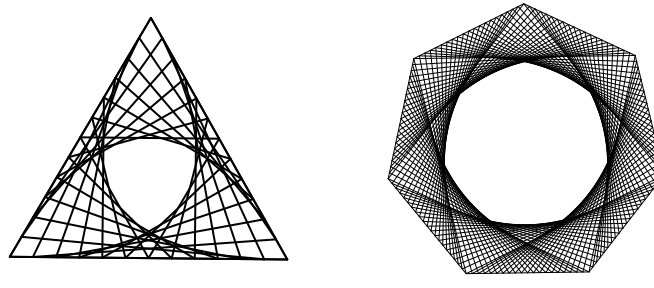


Na okraji obrazce vzniká „kulatý“ obrys, který má tvar paraboly, přičemž vykreslené úsečky jsou tečny této paraboly. Obrys pochopitelně není „doopravdy kulatý“, protože obrázek je tvořen pouze z rovných úseček.

S tímto vzorem jsme si trochu pohráli a drobnými úpravami jsme dostali řadu zajímavých obrazců. Stačilo třeba měnit úhel a vykreslit vzor opakovaně, abychom dostali zajímavé variace na hvězdy:

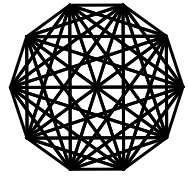


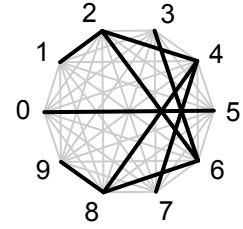
Dál jsem nechal želvy chodit za sebou v mnohoúhelníku. Jedna želva dostala náskok, druhá šla za ní a po každém kroku jsme vykreslili spojnici. Želvy chodily záměrně pomalu, každou hranu mnohoúhelníku si rozložily třeba na deset malých krůčků.



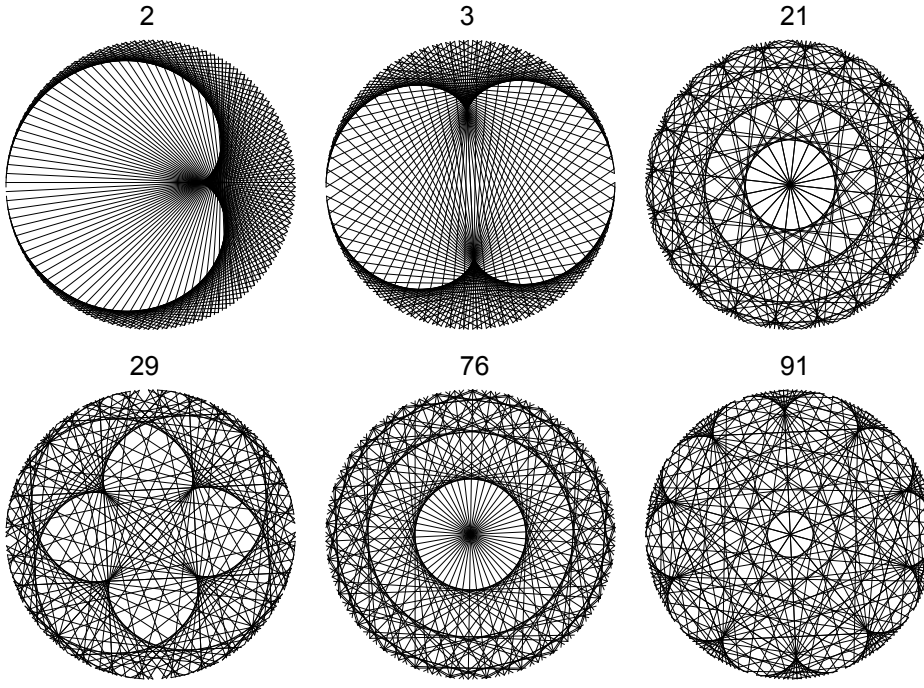
Pomocí spojnic můžeme také snadno vykreslit mnohoúhelník se všemi uhlopříčkami. Jedna želva jde „pomalu“, zastaví vždy v jednom vrcholu, nechá druhou želvu oběhnout celý mnohoúhelník a s její pomocí vykreslí všechny uhlopříčky z daného vrcholu.

Překvapivě zajímavé vzory jsme dostali, když jsme experimentovali s různými způsoby, jak vykreslit jen některé uhlopříčky N-úhelníku. Nechal jsem dvě želvy obcházet vrcholy, přičemž jedna z nich šla k-krát rychleji než druhá, a po každém posunu jsme nechali vykreslit spojnici. Pokud například vezmeme desetiúhelník a jedna želva jde dvakrát rychleji než druhá, dostáváme:



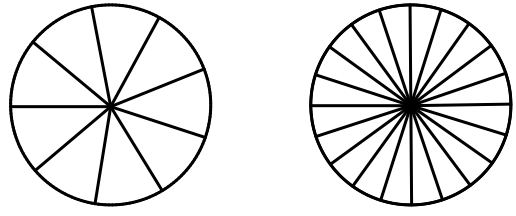


To nevypadá zas tak zajímavě. Ale když vezmeme větší N a dobrou volbu k, dostáváme velmi zajímavé obrázky. Například pro $k = 2$ vznikne na obrázku obrys křivky srdcovky, na kterou jsme již narazili dříve. Několik příkladů pro $N = 200$:



Výzvy

Loukoťové kolo ⋆⋆
Za využití dvou želv vykreslete loukoťové kolo:



Variace na honičky ⋆⋆⋆

Vyzkoušejte různé variace na honičky:

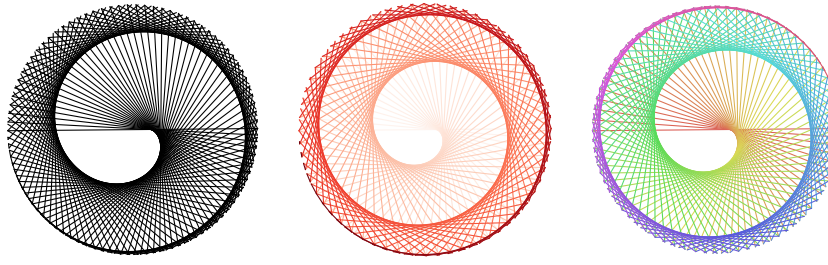
- Více želv honí jednu (první ukázka v textu) za využití různých startovních pozic (náhodně, v řadě, ve čtverci), různých tras unikající želvy (rovně, cik-cak, dokolečka), různé rychlosti želv (konstantní, odstupňované, náhodné).
- Cyklická honička (druhá ukázka v textu) opět za využití různého nastavení.
- Místo pronásledování udělat „únik" – želva se naopak snaží od vybrané jiné želvy utéct na druhou stranu.
- Honička a únik: Každá želva si z ostatních vybere jednu kořist a jednoho lovce. Kořist se snaží dohonit, současně však utíká před lovcem.

Experimenty s uhlopříčkami ⋆⋆⋆

Experimentujte s postupem popsaným v textu pro různé hodnoty N, k. Zkuste systematicky prozkoumat různé varianty a hledat zajímavé vzory.

Závody po kruhu ⋆⋆⋆

Vykreslete následující obrázky a další podobné variace. Základní myšlenka je následující: Dvě želvy umístíme na protější konce kružnice; obě želvy jdou po kružnici, jedna z nich jde rychleji; po každém kroku vykreslíme spojnici.



9 Želva a umění

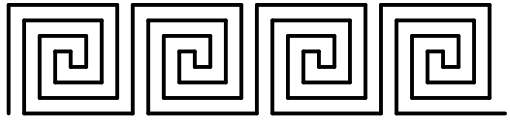
Nakonec jsme se s Žofkou pustili do průzkumu umění. Želvy žijí dlouho a Žofka za svůj život procestovala velkou část světa. Vyprávěla mi o různých obrazech a vzorech, které v různých koutech světa viděla, a toužila zkusit si některé z nich vykreslit. Tak jsme se do toho pustili a při vykreslování jsme využili mnohé z toho, co jsme se před tím spolu naučili.

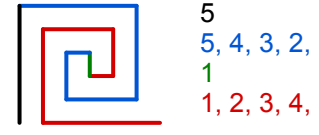
Tato závěrečná část mého vyprávění o našich obrázkových dobrodružstvích má jinou podobu než ostatní kapitoly. Tato kapitola neobsahuje samostatné výzvy, spíš ji lze celou chápat jako jednu velkou výzvu. Zmiňuji několik typů vzorů, které jsme s Žofkou zkoušeli vykreslovat, ukazuji vždy pár obrázků pro inspiraci a zmiňuji rady, jak máte k vykreslování přistupovat. Neuvádím ale kompletní řešení, vykreslení zmíněných obrázků představuje první výzvu, do které se můžete pustit.

Hlavní výzvou je však hledání vlastních námětů a jejich zpracování. Zkuste hledat další náměty v knížkách o umění, na internetu anebo prostě jen dobrým pozorováním světa kolem sebe.

Meandry

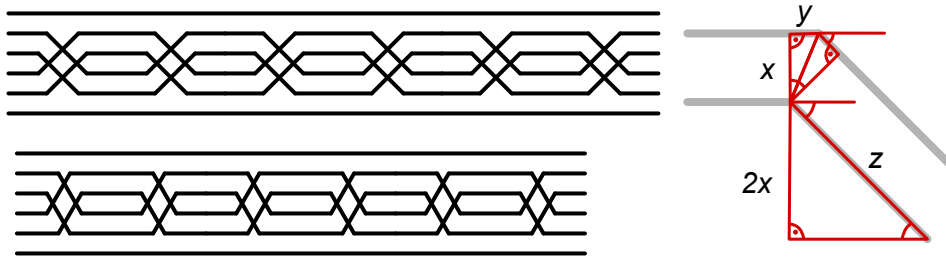
Meandr je vzor tvořený z klikaté čáry, která se pravidelně opakuje. Meandry jsou známé především z umění antického Řecka a Říma, ale podobné vzory se vyskytují i v mnoha dalších kulturách. Nejznámější meandr je ve tvaru čtvercové spirály. Zde je celkem jasné, jak vzor vykreslovat, jen si musíme dobře rozmyslet, jak se střídají délky jednotlivých úseků, což je trochu komplikovanější, než se na první pohled může zdát:





Pokud se pustíme do vzorů, které používají i jiné úhly než 90 stupňů, musíme se trochu víc zamyslet nad úhly a vzdálenostmi. Pro rozmýšlení se hodí nakreslit si vzor ve velkém a dokreslit si do něj pravoúhlé trojúhelníky. Třeba v následujícím vzoru jsou klíčové trojúhelníky s úhly 90°, 60° a 30° (dva z nich jsou vyznačeny na obrázku, lze jich tam vkreslit ještě řadu dalších). Pro určení vzdáleností, o které se má želva posunovat, si stačí ujasnit, že v tomto trojúhelníku má kratší odvěsna poloviční délku než přepona.

U některých obrazců má smysl mít úhel jako volitelný parametr, který ovlivňuje vzhled výsledného meandru. U takových vzorů už k výpočtu vzdáleností potřebujeme využít goniometrické funkce. Následující obrázek má volitelnou velikost zatáčení α, ukázky jsou pro 45° a 60°.



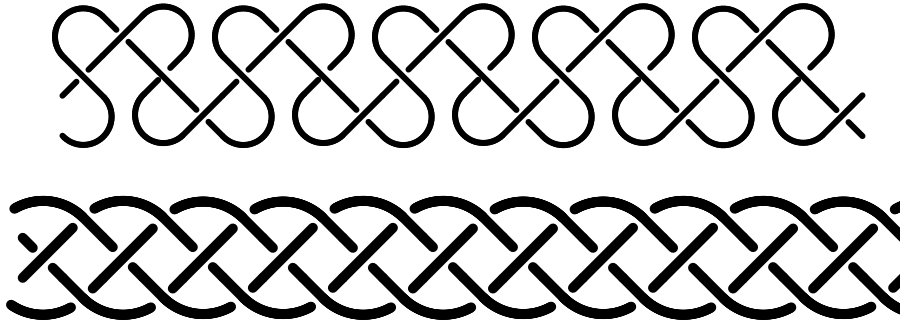
Základ řešení spočívá opět v tom, že si do obrazce zakreslíme pravoúhlé trojúhelníky a v nich vyznačíme úhly – v obrázku jsou vyznačeny výskyty úhlu α. Pomocí pravoúhlých trojúhelníků pak dopočítáme potřebné vzdálenosti. Vzdálenost x je rozestup mezi čárami, to je vstupní parametr našeho vzoru. Z pravoúhlých trojúhelníků pak určíme $y = x \cdot \tan(\alpha/2), z = 2x/\sin(\alpha)$. Za využití těchto vzdáleností pak již vyskládáme celou trasu pro želvu.

„Tohle mi přijde jako moc práce a málo zábavy," začala si najednou stěžovat Žofka. „Není to jednoduché vymyslet, a přitom nakonec vykreslím jen starý známý vzor."

„Ona trocha přemýšlení neuškodí. A navíc, když už vymyslíme program pro základní vzor, můžeme s ním kreativně experimentovat," povzbuzoval jsem ji. „Můžeme třeba zkoušet vykreslování s různými parametry nebo přidat do programu zatočení navíc. Dostaneme tak velmi snadno zajímavé variace na klasické vzory."

Proplétané vzory

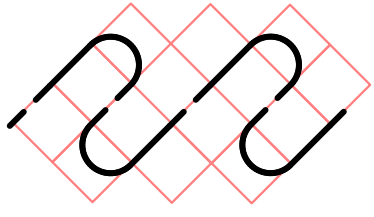
Další podobná kategorie jsou „proplétané vzory“. Ty jsou známé především z keltského umění, ale opět se vyskytují i v umění mnoha dalších kultur. Optického efektu „proplétání“ dosáhneme u želví grafiky pomocí zvedání pera – blízko bodu křížení čar zvedneme pero a přerušíme vykreslování jedné z čar. Pro lepší vzhled také použijeme tlustou čáru.

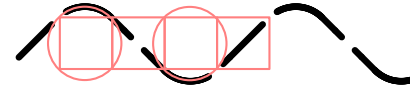


„Tady ty vzory vypadají na první pohled komplikovaně. Na to bude určitě potřeba složitý program,“ strachovala se Žofka.

„Neměj zbytečné obavy. Proplétané vzory se často skládají z mnoha čar, které mají ale celkem jednoduchý tvar. Jen je potřeba obrázek správně rozplést,“ uklidnil jsem ji.

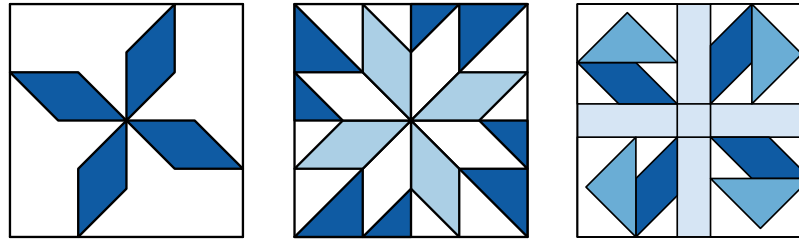
Abychom obrázek mohli „rozplést“, vyplatí se nakreslit si pod vzor pravidelnou čtverečkovou mřížku a zaměřit pozornost na vykreslení pouze jedné z čar. Třeba u uvedených příkladů tak zjistíme, že vzory se vlastně skládají z jednoduché posloupnosti rovných čar a oblouků (půlkruhový či čtvrtkruhový oblouk), jedinou komplikací tak zůstává správné zvedání a pokládání pera. Poté, co vykreslíme jednu čáru, potřebujeme želvu vrátit zpět na začátek a vykreslit další čáry. K tomu se hodí využít příkazů „ulož polohu“ a „obnov polohu“ (viz strana 53).





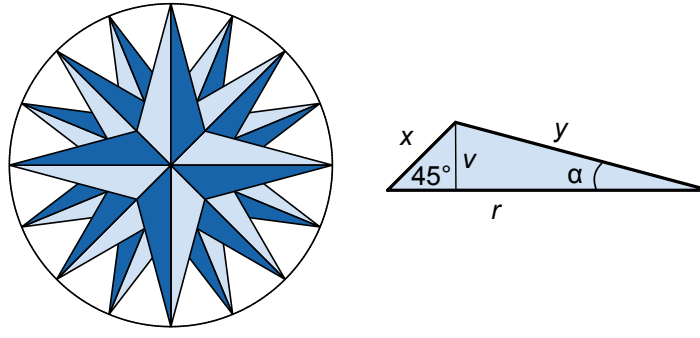
Quilt

Quilt je prošívaná deka se zdobným vzorem tvořeným technikou patchwork (sešívání malých různě barevných kousků látek). Používané vzory jsou často geometrické. Oblíbené jsou například vzory tvořené z čtverců a kosočtverců:



Pro vykreslování želví grafikou je potřeba si především vypočítat délku hrany vnitřních částí v závislosti na délce vnějšího čtverce, k čemuž využijeme Pythagorovu větu. Zbytek je vcelku přímočarý, například úhly jsou všude násobky 45 stupňů.

Další oblíbeným vzorem jsou různé variace na hvězdy, třeba následující:



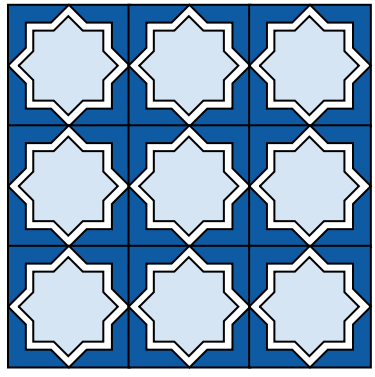
Při jejím vykreslování spočívá hlavní problém ve správném vykreslení jednoho trojúhelníku. Délka dlouhé strany odpovídá poloměru opsané kružnice r, ostrý uhel α je nastavitelný, podle jeho volby dostáváme různě špičaté hvězdy. Zbytek trojúhelníku už je těmito parametry dán a musíme jej dopočítat. Zbývající úhly jsou 45° a $135° - \alpha$. Ke vzdálenostem dostáváme z pravého pravoúhlého trojúhelníku $v = y\sin(\alpha)$, $r - v = y\cos(\alpha)$. Po sečtení a úpravě dostáváme $y = r/(\sin(\alpha)+\cos(\alpha))$. Z levého pravoúhlého trojúhelníku pak $x = \sqrt{v^2 + v^2} = v\sqrt{2} = y\sin(\alpha)\sqrt{2}$.

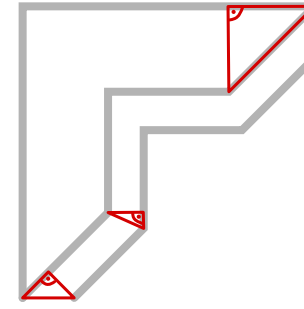
Jakmile už umíme správně udělat trojúhelník, už ho jen opakovaně vykreslujeme ze středové pozice ve vhodných rotacích a ve správném pořadí od „spodních“ (překrytých) po „horní“ (nepřekryté).

Islámské geometrické vzory

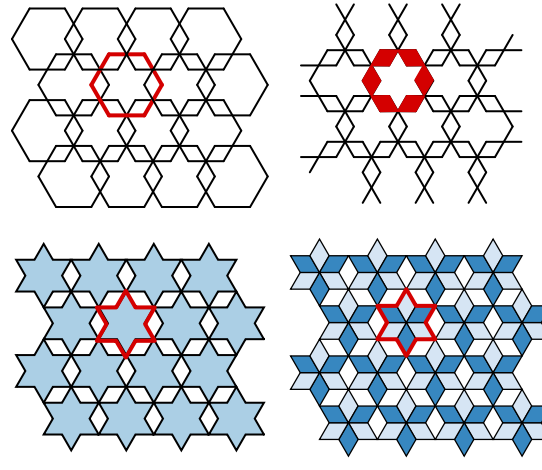
Islámské umění a architektura často využívají podivuhodné geometrické vzory, které mohou posloužit jako bohatý zdroj inspirace na zajímavá cvičení s želví grafikou. Podíváme se na variace na základní motivy, jež se v islámském umění často vyskytují.

Relativně jednoduchý, ale vizuálně zajímavý vzor dostaneme ze čtvercových dlaždic, do kterých je vkreslena osmicípá hvězda. Při vykreslování musíme hlavně správně určit délky stran, k tomu nám opět pomůže zvýraznit si klíčové pravoúhlé trojúhelníky.

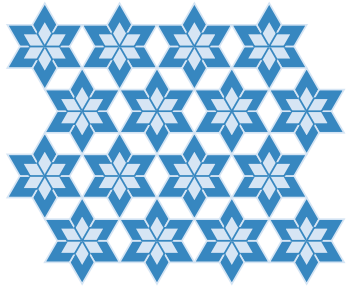


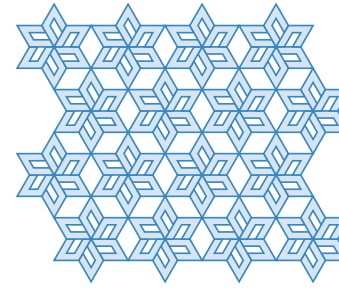


Překvapivě zajímavý vzor dostaneme drobnou úpravou šestiúhelníkové teselace („dlaždičkování“). Na tento vzor se můžeme dívat mnoha různými způsoby, například jako na překrývající se šestiúhelníky, vyskládané kosočtverce nebo šesticípé hvězdy. Každý z těchto pohledů vede na jiný způsob vykreslování pomocí želví grafiky.



Jakmile máme hotové vykreslování základního vzoru, stačí jen mírně upravit vykreslování dílčích prvků a dostaneme řadu zajímavých variací.

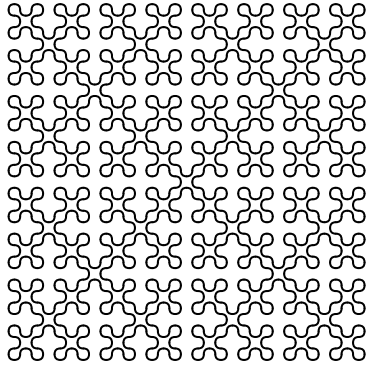


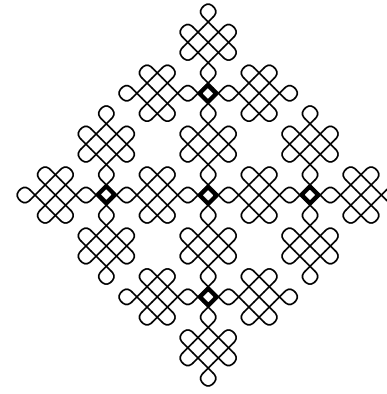


Indické kolamy

„Mám pro tebe jeden obzvláště zajímavý námět," povídá jednoho dne Žofka. „Kdysi dávno jsem byla v Indii a viděla jsem tam takové zajímavé obrazce – říkají jim kolamy. Teď mě napadá, že některé z těch kolamů vypadaly docela podobně jako fraktály, které jsme spolu vykreslovali."

Prozkoumal jsem to a Žofka měla pravdu. Kolamy jsou tradiční indické obrázky kreslené křídou nebo rýžovou moukou. Skládají se z propletených křivek a teček, mnohdy mají komplikovanou geometrickou strukturu a některé z nich mají výrazně rekurzivní podobu. Zaměřili jsme se s Žofkou právě na ty rekurzivní:

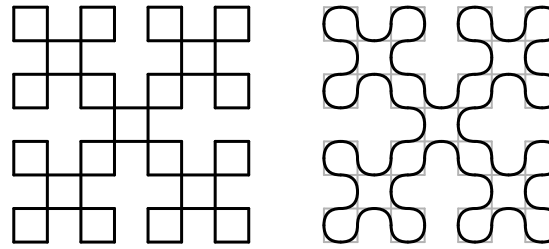




Vykreslování kolamů jsme zkoušeli dělat jak za využití rekurzivních příkazů, tak pomocí L-systémů. Základní strukturu prvního kolamu jsme vykreslili pomocí následujících pravidel:

axiom: F+XF+F+XF

pravidlo:
X ⇒ XF-F-F+X
F+F+XF
úhel: 90°



Aby výsledek vypadal pěkně, potřebovali jsme čáry zaoblit. K tomu bychom mohli využít přístup, který jsme používali pro vykreslování křivek dříve – želva by dělala hodně malých kroků a malých zatáček. To by nám tady ale hodně znepřehlednilo pravidla L-systému. Raději jsem Žofku naučil „plynulé plazení" – místo aby chodila přímo rovně a dělala ostré zatáčky, zatáčí plynule. Výsledný obrázek hned vypadá lépe.

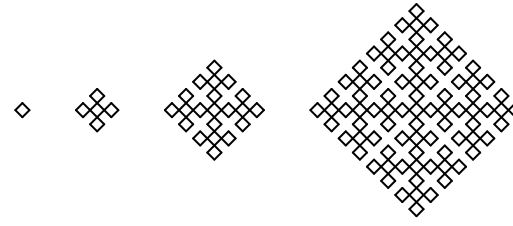
Zmíněné „plynulé plazení" se v počítačové grafice nazývá „kvadratická Bézierova křivka". Bézierovy křivky se pro vykreslování zaoblených křivek využívají v počítačové grafice velice často.

I pro druhý z uvedených kolamů můžeme zachytit základní strukturu jednoduchým L-systémem:

axiom: -X--X

pravidlo:
X ⇒ XFX--XFX

úhel: 45°



Pokud chceme věrně reprodukovat indickou předlohu, hodí se L-systémy rozšířit o „finalizační pravidla". Kolamy často obsahují základní rekurzivní strukturu, která je následně „dozdobena". Klasický L-systém využijeme pro popis rekurzivní struktury, dozdobení pak provedeme pomocí finalizačního pravidla aplikovaného úplně nakonec k doplnění ozdobných prvků. V tomto případě vypadá finalizační pravidlo následovně:

X ⇒ BFA--BFA
A ⇒ F++FFFF--F--FFFF++F++FFFF--F
B ⇒ F--FFFF++F++FFFF--F--FFFF++F

A Želví grafika v Umíme informatiku

Nejjednodušší způsob, jak si vyzkoušet základní práci s želví grafikou, nabízí systém Umíme informatiku (`https://www.umimeinformatiku.cz/`). V tomto systému je želví grafika dostupná ve dvou formách:

- blokové programování
  `https://www.umimeinformatiku.cz/zelvi-grafika`
- textové programování v Pythonu
  `https://www.umimeinformatiku.cz/python-zelva`

Systém nabízí konkrétní zadání s automatickým vyhodnocováním – zadání vždy představuje dílčí obrázek a úkolem je jej vykreslit. Pro každou formu zápisu programů systém nabízí asi 100 konkrétních zadání, která jsou řazena od velmi jednoduchých (základní sekvence příkazů) až po náročnější (proměnné, rekurze). Dostupná cvičení pokrývají zejména základy a soustředí se na procvičení využití cyklů (včetně vnořených cyklů).

B Možnosti realizace

Principy želví grafiky jsou zcela nezávislé na konkrétní technické realizaci. V této příloze je uveden přehled základních možností, jak si želví grafiku vyzkoušet, a jejich výhody a nevýhody. Základní směry realizace jsou poměrně nadčasové, konkrétní detaily se však mohou rok od roku měnit. Tabulka 1 uvádí odkazy na konkrétní implementace dostupné v době vydání knihy.

Většina způsobů realizace využívá anglické příkazy. Tabulka 2 udává přehled hlavních příkazů, které jsou v této knize použity, a jejich standardní anglickou podobu (detaily zápisu se v jednotlivých prostředích mohou drobně lišit).

Programování pomocí grafických bloků

Programování pomocí grafických bloků je pro úplné začátečníky nejjednodušší možnost, jak začít s želví grafikou a programováním obecně. Při tomto přístupu se program skládá pomocí přetahování a spojování bloků myší (či prstem na tabletu) a pomocí výběru hodnot parametrů z předdefinované nabídky, tj. není potřeba nic psát na klávesnici a při programování nehrozí syntaktické chyby.

Tento přístup má výhodu, že je velmi intuitivní. Základní programy tímto způsobem zvládnou vytvořit i malé děti. Cenou za tuto intuitivnost je pochopitelně omezený záběr. Tento přístup je vhodný pro základní experimentování v rozsahu úvodních kapitol, pro složitější práci s proměnnými a funkcemi už vhodný není.

Konkrétní realizace dnes většinou staví na technologii „Blockly“, což je obecné prostředí pro programování pomocí grafických bloků ve webovém prohlížeči. Další konkrétní možností je Scratch, což je rozsáhlé prostředí pro programování pomocí grafických bloků, ve kterém lze mimo jiné zkoušet i želví grafiku.

Tabulka 1: Odkazy na konkrétní prostředí zmíněné v textu

Prostředí	Odkaz
Blockly	`https://blockly-games.appspot.com/turtle`
Scratch	`https://scratch.mit.edu/`
OzoBlockly	`http://ozoblockly.com/`
Logo	`http://www.logointerpreter.com/`
NetLogo	`https://ccl.northwestern.edu/netlogo/`
Python turtle	`https://repl.it/languages/python_turtle`
L-systémy	`http://nolandc.com/sandbox/fractals/`

Tabulka 2: Slovníček příkazů

příkaz použitý v textu	standardní podoba v angličtině
dopředu	forward
doleva	left
doprava	right
zvedni pero	pen up
polož pero	pen down
nastav barvu pera	pen color
nastav barvu vybarvování	fill color
nastav tloušťku čáry	pen width
udělej tečku	dot
začni vyplňovat	begin fill
ukonči vyplňování	end fill
ulož pozici	push
obnov pozici	pop
pokud	if
opakuj	repeat, for

Logo a NetLogo

Logo je programovací jazyk pocházející z 60. let 20. století, který je úzce spjat s počátky želví grafiky. Tento jazyk a princip želví grafiky (včetně jména) navrhla výzkumná skupina Seymoura Paperta, který želví grafiku následně zpopularizoval knihou Mindstorms a dalšími aktivitami. Tento jazyk se ve spojení s želví grafikou stále používá, ale jeho syntax je už na dnešní poměry zastaralá, takže jej pro použití ve výuce nelze doporučit.

Na jazyk Logo navazuje nástroj NetLogo, který přirozeně pracuje s rozšířením pro „mnoho želv“. Želví grafiku lze v tomto nástroji snadno realizovat. NetLogo je však známé především jako nástroj pro „modelování pomocí agentů“, což je zajímavý přístup k modelování systémů v počítači. NetLogo obsahuje velmi bohatou sbírku příkladů, jež jsou velmi zajímavé a pedagogicky přínosné, většina z nich však nesouvisí se želví grafikou. Programovací jazyk, který NetLogo používá, vychází z historického jazyka Logo (byť s mnoha úpravami) a také není pro výuku programování ideální.

Python

Mnoho běžně používaných programovacích jazyků má dostupné knihovny pro podporu želví grafiky. Nejběžnější volbou z těchto programovacích jazyků je Python, který je nyní obecně preferovanou volbou pro výuku úvodního programování. Knihovna `turtle` pro želví grafiku je navíc obsažena ve standardní distribuci, takže její použití je obzvláště jednoduché (není třeba nic speciálního instalovat). Dokonce lze želví grafiku v Pythonu používat i přímo ve webovém prohlížeči, viz odkaz v Tabulce 1.

Python je poměrně intuitivní programovací jazyk i pro začátečníky a základní programy jsou v něm celkem snadné. Současně nám tato volba umožňuje realizovat i všechny koncepty pokryté v knize, protože staví na obecném programovacím jazyku.

Vlastní implementace

Pro trochu pokročilé programátory je velmi dobrou volbou napsat si zcela vlastní implementaci želví grafiky. K tomu lze použít libovolný programovací jazyk podle vlastní volby. Pro realizaci grafického výstupu se hodí formát SVG (Scalable Vector Graphics). Jde o vektorový formát, jenž textovou formou popisuje grafické prvky. V rozsahu, který potřebujeme pro realizaci želví grafiky, má formát SVG velmi jednoduchou syntax, takže vytváření obrázků je snadné. Současně nám formát SVG umožňuje s obrázky snadno dále pracovat – můžeme je otevřít v editoru vektorové grafiky (např. volně dostupný Inkscape). Tento přístup byl použit při přípravě této knihy.

Vlastní implementace želví grafiky vede přirozeně na použití objektového programování – představuje dobré úvodní cvičení pro tento programátorský přístup. Objektová implementace se hodí zejména pro příklady s více želvami, které v základních prostředích pro želví grafiku většinou nelze realizovat. S objektovou implementací je jejich realizace přímočará.

Fyzický robot

Představa fyzické želvy kreslící do písku se typicky používá jen jako metafora pro uvedení tématu želví grafiky. Nicméně původní autoři konceptu želví grafiky pracovali i s fyzickým robotem (připomínajícím želvu), který opravu kreslil na papír. Práce s fyzickým robotem po stránce principů nepřidává nic moc nového – programátorské i matematické koncepty zůstávají stejné, jen realizace je prakticky komplikovanější. Nicméně po stránce atraktivity má práce s robotem svoje kouzlo.

Snadno použitelný je například Ozobot – jde o malého robota, kterého lze programovat pomocí varianty prostředí Blockly nazvané OzoBlockly. Robot má fixní konstrukci se dvěma koly, což mu umožňuje základní pohyb po rovině (pohyb dopředu, zatáčení) a může tak snadno realizovat pohyby, které jinak provádíme s virtuální želvou. Robot má navíc senzory, jimiž je schopen rozlišovat barvu papíru, což poskytuje další prostor pro zajímavé náměty. Ozobot je však příliš malý na to, abychom k němu mohli přidělat pero a vykreslovat čáry.

Pokud chceme robota, který bude i kreslit, můžeme využít Lego Mindstorms – rozšíření klasického Lega o programovatelnou kostku, sensory a motory. Programy lze psát pomocí Blockly nebo různých variant běžných programovacích jazyků. Konstrukce robota s přidělanou tužku, kterou kreslí na papír, představuje zajímavou výzvu (inspiraci můžete snadno najít na YouTube).

L-systémy

Specifickou částí knihy jsou L-systémy, s nimiž si ve výše zmíněných realizacích želví grafiky typicky nemůžeme hrát úplně snadno. Při použití obecného programovacího jazyka pochopitelně lze L-systémy naprogramovat. To vyžaduje implementaci přepisovacích pravidel, což je celkem snadné programátorské cvičení, ale pro úplné začátečníky to je bariéra v tom, aby si L-systémy vyzkoušeli.

Je však možné využít prostředí, která jsou zaměřená přímo na L-systémy. Ta umožňují prostě zadat pravidla a nechat si vygenerovat výsledný obrázek, bohužel však bez hlubšího propojení s ostatními prvky želví grafiky. Jednou z takových možností je využít program pro vektorovou grafiku Inkscape, který obsahuje podporu pro L-systémy jako vestavěné rozšíření. Výhodou je, že s výstupem pak můžeme dále (manuálně) pracovat. Dále existuje řada

webových stránek umožňujících experimentování s L-systémy v prohlížeči (viz příklad uvedený v Tabulce 1).

C Pedagogické a historické poznámky

Tato příloha je určena učitelům, kterým nabízí shrnutí pedagogických aspektů želví grafiky a dává tipy k zapojení do výuky.

Pedagogické výhody želví grafiky

Želví grafika představuje dobrý způsob, jak si zajímavým způsobem vyzkoušet programování a procvičit si mnoho pojmů z matematiky. Základní výhodou želví grafiky je intuitivnost a atraktivnost. Oproti jiným způsobům vykreslování obrázků, konkrétně ve srovnáním s vykreslováním za využití kartézských souřadnic, má želví grafika výhodu intuitivní představy pohybující se želvy. Postup vykreslování si snadno můžeme „vyzkoušet na sobě". To neznamená, že by všechny programy pro želví grafiku byly jednoduché – v této knize je uvedena řada námětů, u nichž se může zapotit i dobrý programátor. Ale díky intuitivnosti je snadné začít a postupně se zlepšovat.

Želví grafika nabízí přirozenou ilustraci mnoha zdánlivě abstraktních pojmů z matematiky. Jde především o geometrické pojmy: práci s úhly, počítání vzdáleností, Pythagorovu větu, goniometrické funkce, fraktály. Pomocí želví grafiky můžeme ale ilustrovat i koncepty, které nemají přímočarou „obrázkovou interpretaci": soudělnost, nejmenší společný násobek, aritmetickou a geometrickou posloupnost. U programování pak lze želví grafiku využít od úplných základů (úvodní seznámení s programováním), až po pokročilá témata, jako je rekurze a objektově orientované programování.

Klíčovou výhodou želví grafiky jsou obrázkové výstupy. Pro většinu lidí je tvorba obrázků výrazně zajímavější než jiné aplikace programování – obrázek spirály potěší víc než správně vypočítaný faktoriál z deseti. I jednoduchým programem lze vytvořit pěkné obrázky, což je pro mnoho studentů zdrojem přirozené vnitřní motivace. Obrázky také poskytují bezprostřední zpětnou

vazbu – student při tvorbě programu hned vidí, co program dělá a zda je na dobré či špatné cestě.

Želví grafika a výuková témata

Želví grafiku lze využít k procvičení mnoha oblastí. Následuje přehled těch nejtypičtějších se stručným komentářem k cílové skupině, vhodnému implementačnímu prostředí a s odkazy na nejrelevantnější kapitoly textu.

Základoškolská geometrie

Nejjednodušší využití želví grafiky je pro procvičení základních geometrických pojmů, jako jsou úsečky, úhly, vzdálenosti. Za tímto účelem můžeme minimalizovat „programátorské aspekty" želví grafiky a soustředit se pouze na základní příkazy pro pohyb. Vhodné je využít jednoduché grafické prostředí, např. Blockly. Náměty na příklady nabízí kapitoly 2 a 3.

Základy programování

Další základní využití želví grafiky spočívá v představení a procvičení základních programátorských konstrukcí, jako jsou proměnné, cykly a funkce. Želví grafika je ideální pro úvodní seznámí s programováním. Během pár minut jsou studenti schopni vytvořit programy, které dělají zajímavé věci, a většina programátorských konceptů je v kontextu želví grafiky intuitivní. Na prvním stupni ZŠ je vhodné využít grafické prostředí (Blockly). Starší studenti mohou také začít v grafickém prostředí, ale je reálné začít i rovnou v Pythonu. K tomuto využití jsou relevantní především kapitoly 3, 4 a 5.

Středoškolská geometrie

Želví grafika je vhodná i pro vysvětlení a procvičení pokročilejších geometrických pojmů – Pythogorova věta, goniometrické a cyklometrické funkce. Zde můžeme vystačit i s poměrně základními příkazy pro pohyb, hodí se nicméně využít textové prostředí (např. Python), které umožní snadno pracovat s matematickými funkcemi (např. výpočet odmocniny a goniometrických funkcí). K tomuto využití je relevantní především kapitola 6, další rozšiřující náměty jsou pak v kapitole 9.

Dělitelnost

Netypické, ale zajímavé téma k prozkoumání pomocí želví grafiky představuje dělitelnost (dělitelnost dvou čísel, soudělnost, nejmenší společný násobek, zbytek po dělení…). Toto téma vyžaduje, aby studenti zvládli základy programování včetně využití proměnných. Prozkoumávání námětů na základní programování, které mají číselné vstupy, pak často přirozeně

souvisí s dělitelností – vzhled výstupních obrázků často závisí na soudělnosti vstupních parametrů. Detailněji je jeden takový námět rozebrán v kapitole 5.

Rekurze

Rekurze je důležitý koncept v programování a informatice obecně. Její použití je na jednu stranu velmi jednoduché, na druhou stranu jde o myšlenkově hodně náročný koncept. Želví grafika ve spojení s fraktály poskytuje velmi dobrý prostor pro získání vhledu do toho, jak rekurze funguje. Využití rekurze je relevantní až od úrovně středoškolských studentů a vyžaduje využití textového prostředí pro zápis programů (doporučeno je použít Python). Tímto tématem se zabývá detailně kapitola 7.

Objektově orientované programování

Zájemcům o procvičení pokročilejších programátorských témat nabízí želví grafika přirozené procvičení objektově orientovaného programování, tentokrát ale úplně jiným stylem než u ostatních témat – neprocvičujeme objekty v želví grafice, ale skrze vlastní implementaci želví grafiky. Tento směr realizujeme skrze cvičení v běžných programovacích jazycích (krom Pythonu může být přirozené použít například Javu, která je silně objektově orientovaná).

Vlastní implementace základní verze želví grafiky je poměrně přímočará a představuje vhodné cvičení i pro úplně základní pojednání o objektově orientovaném programování. Zajímavým cvičením pak je rozšíření na „více želv“ (kapitola 8).

Způsob výuky

Počátky želví grafiky jsou spojeny se Seymourem Papertem, který prosazoval konstruktivistické pojetí výuky a želví grafiku představoval především jako konkrétní ukázku obecných konstruktivistických principů. V principu však lze želví grafiku použít širokou škálou různých způsobů od čistě konstruktivistického až po čistě frontální výklad, přičemž ani jeden z extrémů není ideální.

Želví grafika se typicky využívá ve spojitosti s výukou některého z výše uvedených témat. Je tedy vhodné dát studentům alespoň základní výklad o souvisejících pojmech (programátorských i matematických), případně průběžně dávat potřebné nápovědy nebo poradit studentům, jak celkový úkol rozložit na jednodušší úkoly. Zejména u složitějších obrázků nelze očekávat, že studenti samostatným zkoumáním objeví goniometrické funkce nebo princip rekurze.

Na druhou stranu je želví grafika velmi přirozené prostředí pro objevování a kreativitu a byla by škoda použít ji jen jako netypické prostředí pro standardní výklad. Rozhodně je vhodné studenty povzbuzovat k experimentování, například formou vykreslování vlastních obrázků. Obzvláště vhodná je želví grafika pro prozkoumávání otázek typu „Co když …“. Co když trochu změním úhel? Co když změním směr otáčení? Co když změním počet opakování cyklu?

Zdroje

Abelson, H. & DiSqessa, A. A. (1986). Turtle geometry: The computer as a medium for exploring mathematics. MIT press.

Beyer, J. (2009). The Quilter's Album of Patchwork Patterns: More Than 4050 Pieced Blocks for Quilters. Breckling Press.

Bourgoin, J. (2012). Arabic geometrical pattern and design. Courier Corporation.

Brusilovsky, P., Calabrese, E., Hvorecky, J., Kouchnirenko, A. & Miller, P. (1997). Mini-languages: a way to learn programming principles. Education and Information Technologies, 2(1), 65–83.

Caspersen, M. E. & Christensen, H. B. (2000). Here, there and everywhereon the recurring use of turtle graphics in CS 1. In ACM International Conference Proceeding Series (Vol. 8, pp. 34–40).

Ďurčeková, Z. (2022). Interpretable clustering of turtle graphics programs.Bakalářská práce. Masarykova univerzita, Fakulta informatiky.

Jones, O. (1868). The grammar of ornament. B. Quaritch.

Kelleher, C. & Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. ACM Computing Surveys, 37(2), 83–137.

Newell, B. (1988). Turtle Confusion: Logo Puzzles and Riddles: Barry Newell. Curriculum Development Centre.

Papert, S. (1980). Mindstorms: Children, computers, and powerful ideas. Basic Books, Inc.

Prusinkiewicz, P., Hammel, M., Hanan, J. & Mech, R. (1996). L-systems: from the theory to visual models of plants. In Proc. of the 2nd CSIRO Symposium on Computational Challenges in Life Sciences (Vol. 3, pp. 1-32).

Prusinkiewicz, P. & Lindenmayer, A. (2012). The algorithmic beauty of plants. Springer Science & Business Media.

Yanagisawa, K. & Nagata, S. (2007). Fundamental study on design system of kolam pattern. Forma, 22(1), 31–46.