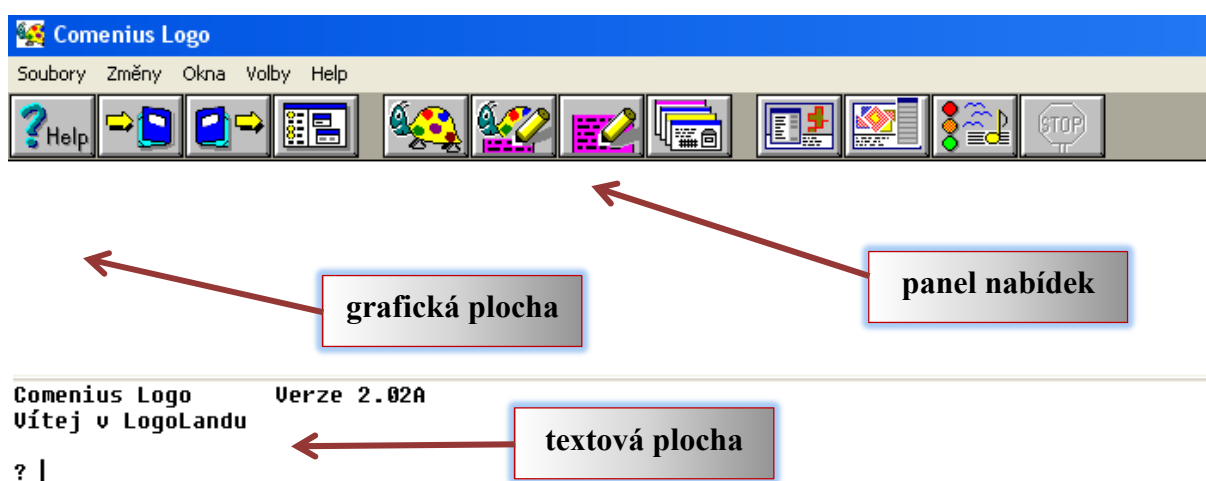


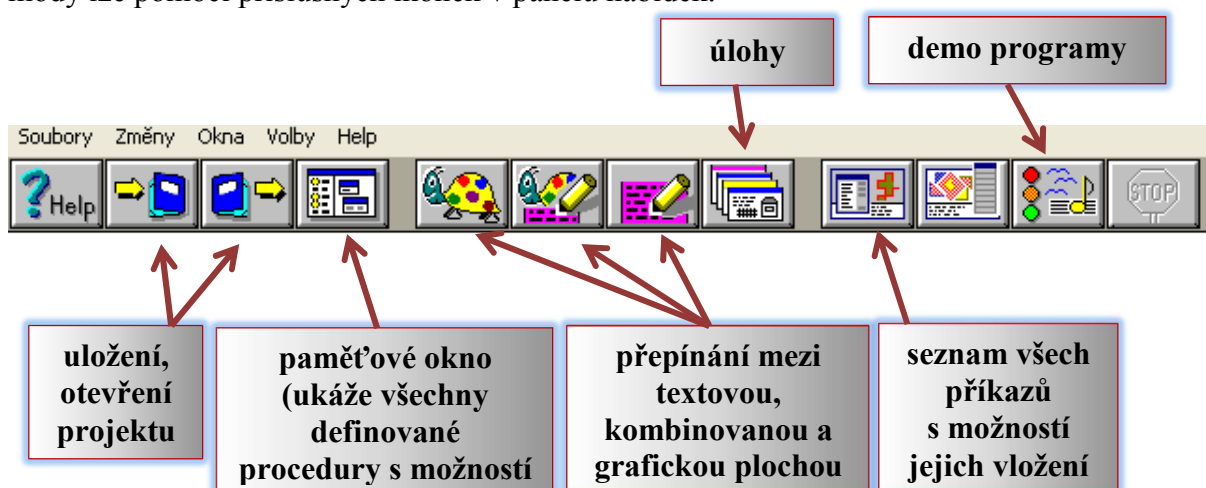
Programování v jazyku LOGO - úvod

Programovací jazyk LOGO je určen pro výuku algoritmizace především pro děti školou povinné. Programovací jazyk pracuje v grafickém prostředí, přičemž jednou z jeho podstatných vlastností je možnost práce s tzv. želví geometrií, která nepotřebuje souřadný systém a přesto umožňuje velmi krásné geometrické manipulace.


Jaký je princip želví geometrie? Uživatel řídí malý objekt (želvu) v rovině (přesněji řečeno na monitoru počítače). Želva reaguje na několik jednoduchých příkazů - **forward**, který posune želvu o daný počet jednotek, **right**, který pootočí želvu na místě doprava o daný počet stupňů, a příkazy opačné - **back** a **left**. Stručně řečeno, želva si sama pamatuje svoji polohu (která se mění použitím forward a back) a směr (ten se mění použitím left a right). Navíc má želva pero, kterým při pohybu kreslí, toto pero se dá zvednout příkazem **penup**, a spustit pro kreslení příkazem **pendown**.

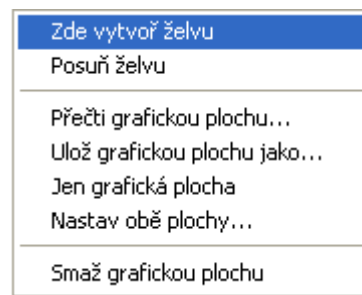


Základní okno programu má nahoře standardní panel nabídek a plochu, která může zobrazovat grafickou, textovou nebo grafickou i textovou plochu. Přepínat mezi jednotlivými mody lze pomocí příslušných ikonek v panelu nabídek.



Než začneme, zvolíme si podobu želvy. Je to jakýsi ukazatel, který naznačuje, ve které části grafické plochy se právě nacházíme. Pomocí pravého tlačítka rozbalíme dialogové okno a

zvolíme možnost „Zde vytvoř želvu“. V dalším dialogu si potom zvolíme obrázek, který bude želvu zastupovat, zvolíme souřadnice (pro začátek ideální obě nulové) a také zadáme krátké jméno (bez české diakritiky!). Vybereme si např. znak šipky.  V této chvíli jsme připraveni napsat první jednoduchý program.



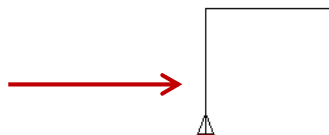
Poznámka: Zvolená podoba „želvy“ nijak neovlivňuje psaní programů, může např. zpříjemňovat uživateli práci s programem.

Program začneme psát v textové části. Každý řádek je uvozen znakem „?“

Použití základních příkazů

Můžeme začít s kreslením. Pro nakreslení čtverce o straně 100 jednotek stačí použít příkazy **forward**.

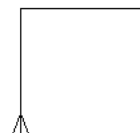
```
? forward 100 right 90
? forward 100 right 90
? forward 100 right 90
? forward 100 right 90
```



Pro začátek si stačí všimnout, že želva se dostala zpět na původní místo. Existuje možnost, jak v jazyku LOGO lépe popsat opakování příkazů. Ještě předtím ovšem můžeme smazat obrazovku příkazem **cs** (clear screen).

Protože se část sekvence příkazů 4x opakuje, můžeme zadat opakování pomocí příkazu **repeat**:

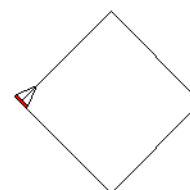
```
? repeat 4 [forward 100 right 90]
```



Jednou z velkých výhod želví geometrie je, že čtverec lze nakreslit v libovolné poloze.

Zkusme následující sekvenci příkazů:

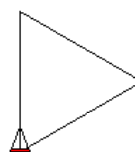
```
? cs right 45 repeat 4 [forward 100 right 90]
```



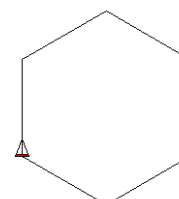
Poznámka: Abychom nemuseli každou sekvenci příkazů znovu vypisovat, ukládají se všechny dosud zadané příkazy v paměti. Vyvolat (nalistovat) je lze pomocí šipky nahoru nebo dolů na klávesnici.

Podobným způsobem můžeme nakreslit trojúhelník nebo např. šestiúhelník:

```
? repeat 3 [forward 100 right 120]
```



```
? repeat 6 [forward 100 right 60]
```



Jak vytvořit novou proceduru

Abychom stále nemuseli používat příkaz repeat pro konstrukci čtverce, můžeme si tuto konstrukci nazvat jménem, např. čtverec. V programování se říká, že vytvoříme **proceduru**. K tomu je v jazyku LOGO příkaz **to** ukončený slovem **end**.

```
? to čtverec  
> repeat 4 [forward 100 right 90]  
> end  
Nadefinoval jsi čtverec
```

Poznámka: Všimněme se, že v průběhu zadávání procedury jsou začátky řádku uvozeny znakem „>“, který nás upozorňuje, že jsme ve stadiu tvorby procedury. Po ukončení definice procedury se objeví hláška o vytvořené proceduře.

Pokud vyzkoušíme práci s touto procedurou, napadnou nás dvě věci. Naučili jsme želvu nový příkaz (umí reagovat na příkaz – proceduru -čtverec). Chtělo by to vylepšení, zatím můžeme kreslit pouze čtverce o velikosti strany 100. Proč by nemohl příkaz čtverec fungovat stejně jako forward? To je tak, že bychom velikost zadali až před samotným nakreslením? Proceduru upravíme následovně:

```
? to čtverec :a  
> repeat 4 [forward :a right 90]  
> end  
Nadefinoval isi čtverec
```

jméno proměnné je vždy za dvojtečkou

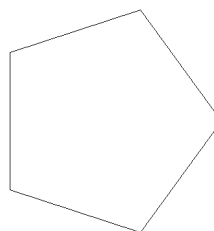
na proměnnou se odkazujeme v těle procedury

Poznámka: Nesmíme zapomenout napsat dvojtečku před jménem proměnné (která symbolizuje velikost strany čtverce). Pro vykreslení čtverce o straně 50 jednotek nyní použijeme příkaz čtverec 50.

Podobně bude vypadat procedura pro vykreslení pravidelného pětiúhelníku:

```
? to pětiúhelník :a  
> repeat 5 [forward :a right 72]  
> end  
Nadefinoval jsi pětiúhelník
```

```
? pětiúhelník 200
```



Poznámka: Velikost úhlu volíme tak, aby celkový součet všech úhlů dal vždy celkem 360°.

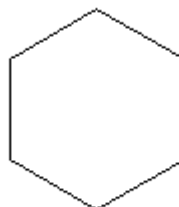
Cvičení

1. Napište proceduru **trojúhelník :a** pro nakreslení rovnostranného trojúhelníka se stranou délky **:a**.
2. Napište proceduru **šestiúhelník :a** pro nakreslení pravidelného šestiúhelníku se stranou délky **:a**.
3. Napište proceduru **desetiúhelník :a** pro nakreslení pravidelného desetiúhelníku se stranou délky **:a**.

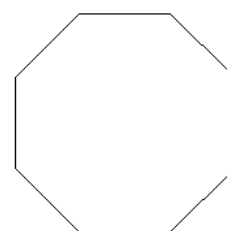
Zkusme napsat proceduru **n.úhelník :n :a**, která nakreslí pravidelný n-úhelník s počtem **:n** stran o velikosti každé strany **:a**. Základem klasické opakování s krokem dopředu a zahnutím. Jediným problémem bude stanovit úhel zahnutí. Ten je totiž pro každý pravidelný n-úhelník jiný. Protože ale víme, že součet všech úhlů má být vždy 360° , stačí pro zjištění velikosti úhlu vydělit číslo 360 množstvím stran n-úhelníku. V programovacím jazyku LOGO se matematický výraz uvádí do závorky, násobení je hvězdička, dělení lomítko.

```
? to n.úhelník :n :a  
> repeat :n [forward :a right (360/:n)]  
> end
```

? n.úhelník 6 50



? n.úhelník 8 70

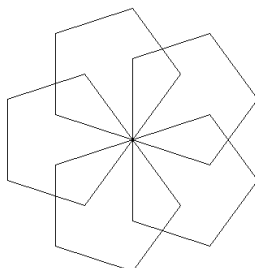


Zkuste předpovědět a posléze vyzkoušet, co vykreslí následující procedura:

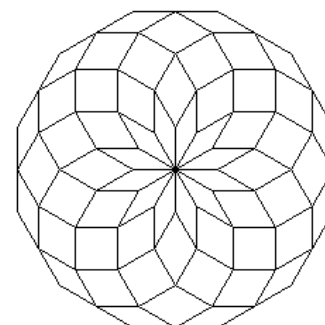
```
? to n.květ :n :a  
> repeat :n [n.úhelník :n :a]  
> end
```

A výsledek?

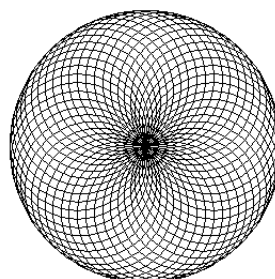
? n.květ 5 100



? n.květ 12 30



? n.květ 50 8



Zkus danou úlohu různě modifikovat.

Kreslení oblouků

Kružnice

Začneme kružnicí, tu totiž můžeme nakreslit jako hodně „hustý“ n-úhelník.

```
? to kružnice  
> repeat 360 [forward 1 right 1]  
> end
```

? kružnice



Poznámka: Berme v úvahu, že 360-ti úhelník musí mít velmi krátkou stranu (v tomto případě má strana velikost jednoho bodu), jinak by se nám nevešel celistvý na grafickou plochu.

Oblouky

Pro kreslení dalších krásných želvích obrázků potřebujeme procedury pro nakreslení části kružnice.

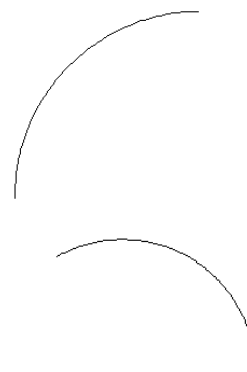
```
? to pravý.oblouk :stupně :a  
> repeat :stupně [forward :a right 1]  
> end
```

? pravý.oblouk 90 3



```
? to levý.oblouk :stupně :a  
> repeat :stupně [forward :a left 1]  
> end
```

? levý.oblouk 120 2

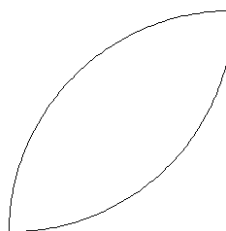


Zkusme teď využít kruhových oblouků k vytvoření procedury **květina**.

Začneme procedurou lístek:

```
? to lístek :a  
> repeat 2 [levý.oblouk 90 :a left 90]  
> end
```

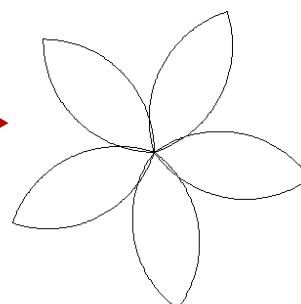
? lístek 4



Pokračujme procedurou květ:

```
? to květ :listky :a  
> repeat :listky [lístek :a left (360/:listky)]  
> end
```

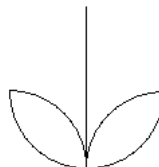
? květ 5 2



A nakonec další pomocná procedura **stonek**:

```
? to stonek :a  
> left 180 forward (120 * :a) left 90 lístek :a left 90 lístek :a  
> end
```

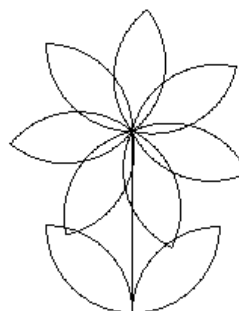
? stonek 1



A nakonec složíme všechny procedury do jediné, a to **květina**:

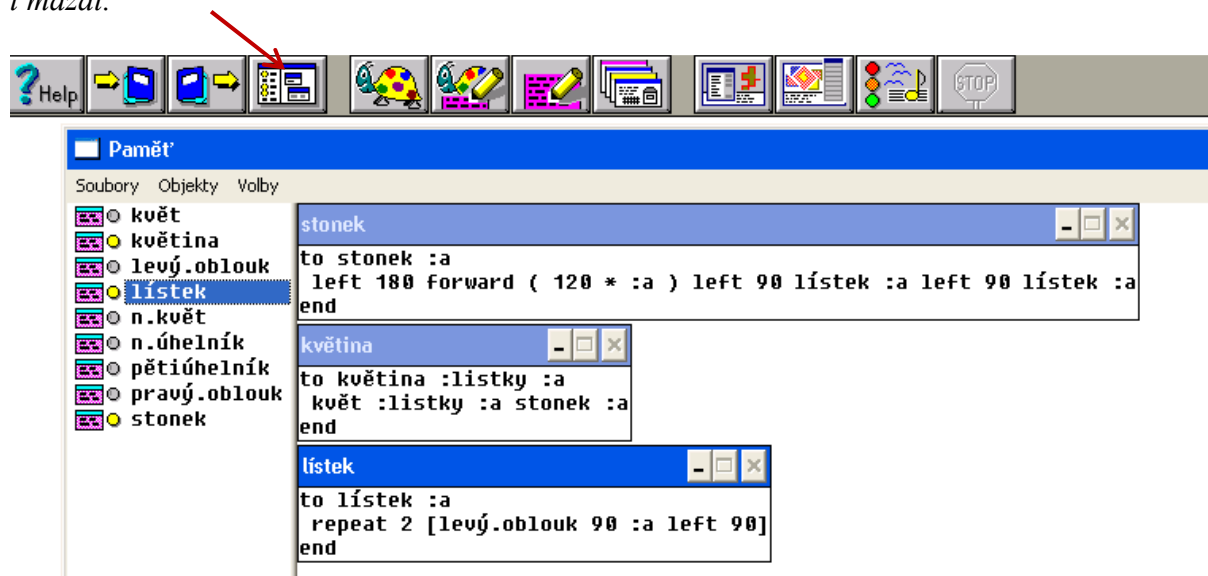
```
? to květina :listky :a  
> květ :listky :a stonek :a  
> end
```

? květina 7 1



*Úkol: Zkuste napsat proceduru **sluníčko :paprsky :a**. kde :paprsky znamenají počet paprsků a :a je velikost sluníčka.*

Poznámka: Pokud se při zadávání procedury spletete a procedura nefunguje, lze každou proceduru upravit pomocí tlačítka v panelu nabídek. Jednotlivé procedury lze tímto způsobem i mazat.



Pozorně prostuduj předchozí text a pokus se odpovědět na otázky k zopakování a pochopení textu:

- 1. Jak vypadá hlavní okno programu LOGO?*
- 2. K čemu programovací jazyk LOGO slouží?*
- 3. Jaké základní příkazy pro pohyb želvy použijeme?*
- 4. Jak se v programu LOGO zadává příkaz opakování?*
- 5. Co je to procedura? Jak ji zadat?*
- 6. Jaký úhel je třeba použít při kreslení pravidelného čtverce, osmiúhelníku, dvanáctiúhelníku?*
- 7. Jak použít proměnnou? Jakým způsobem ji lze použít při zadávání procedury? Proč používat proměnné?*
- 8. Jak v programu LOGO nakreslit kružnici? Jak kruhový oblouk?*
- 9. Lze jednu zadanou proceduru použít v těle (při zadávání) další procedury?*