Úvod do programovacího jazyka Logo

1. Logo: Co to je?

- Jednoduchý programovací jazyk vytvořený pro výuku programování
- Používá tzv. "želvu", která kreslí obrazce na základě příkazů
- Využívá se k vizualizaci algoritmů a rozvíjení algoritmického myšlení

2. Historie Logo

- Vytvořeno v roce 1967 Seymourem Papertem a kolegy
- Původně navrženo pro výuku dětí a jako nástroj pro experimentální učení
- Zaměřeno na rozvoj logického a algoritmického myšlení

3. Význam Logo dnes

- Stále se používá k výuce základů programování
- Zábavná a vizuální forma výuky, ideální pro začátečníky
- Inspirace pro moderní výukové jazyky (např. Scratch)
- https://turtleacademy.com/playground

4. Základní stavební kameny jazyka Logo

- **Pohybové příkazy**: `FORWARD`, `BACK`, `LEFT`, `RIGHT` – umožňují želvě kreslit pohybem
- **Opakování**: `REPEAT` – opakuje skupinu příkazů, např. opakované kreslení čar
- **Procedury**: `TO ... END` – definice vlastních funkcí pro znovupoužití kódu
- **Proměnné**: umožňují používat hodnoty, které se mohou měnit, např. velikost obrazce
- **Podmínky**: `IF` – provádí určité příkazy pouze za určitých podmínek
- **Seznamy**: umožňují ukládat a pracovat s více hodnotami najednou

5. Pohybové příkazy v Logo

- **FORWARD n**: posune želvu dopředu o `n` jednotek
- **BACK n**: posune želvu zpět o `n` jednotek
- **LEFT a**: otočí želvu doleva o `a` stupňů
- **RIGHT a**: otočí želvu doprava o `a` stupňů

6. Příklady pohybových příkazů

- **Příklad 1**: Posuň želvu dopředu o 100 jednotek

```
FORWARD 100
```

- **Příklad 2**: Nakresli čtverec pomocí opakovaných pohybů

```
REPEAT 4 [FORWARD 100 RIGHT 90]
```

7. Kontrola pera

- **PERMIT**, **PENUP** a **PENDOWN**: Příkazy pro kontrolu pera
- **PENUP**: Zvedne pero, želva se pohybuje bez kreslení
- **PENDOWN**: Položí pero, želva zanechává stopu
- **SETPC**: Nastaví barvu pera

8. Nastavení vlastností pera

- **Šířka pera**: Můžeme měnit šířku čáry pomocí příkazu `SETWIDTH`
- **Příklad**: Nastavení šířky pera na 3 a barvy na červenou

```
SETWIDTH 3
setcolor 8
FORWARD 100
```

- Pomocí těchto příkazů můžeme vytvářet rozmanitější kresby

9. Co je procedura?

- **Procedura** je sada příkazů, kterou můžeme pojmenovat a znovu použít
- Pomáhá nám zjednodušit a zorganizovat náš kód
- Vytváří strukturu a umožňuje nám opakovaně používat stejné postupy

10. Definice procedury

- Procedura začíná klíčovým slovem `TO` a končí `END`
- Například procedura pro kreslení trojúhelníku:

```
TO triangle
  REPEAT 3 [FORWARD 100 RIGHT 120]
END
```

- Proceduru zavoláme jednoduše jejím jménem, například: `triangle`

11. Příklady procedur

- **Příklad 1**: Definuj proceduru pro čtverec

```
TO square
  REPEAT 4 [FORWARD 100 RIGHT 90]
END
```

- **Příklad 2**: Použij proceduru k nakreslení více tvarů

```
square
FORWARD 150
triangle
```

12. Proměnné v Logo

- **Proměnné** slouží k ukládání hodnot, které se mohou měnit
- Pomáhají nám vytvářet flexibilnější a obecnější kód
- **Příklad použití proměnné**:

```
TO square :size
  REPEAT 4 [FORWARD :size RIGHT 90]
END
```

- Při volání procedury můžeme určit velikost: `square 50` nebo `square 150`

Náhodný Vstup v Logo

- V Logu můžeme využívat náhodné hodnoty k ovlivnění chování želvy.
- Náhodný vstup umožňuje vytvořit zajímavé a nečekané vzory.
- Používá se příkaz `RANDOM`, který vrací náhodné celé číslo menší než zadaná hodnota.

Příklad: Kreslení čáry o náhodné délce mezi 0 a 100 jednotkami

```
FORWARD RANDOM 100
```

Příklad: Náhodná otočení želvy

- Pomocí příkazu `RANDOM` můžeme želvě dát náhodné příkazy k otočení.
- To vytvoří nepředvídatelné a zajímavé tvary.

Příklad: Želva se otočí o náhodný úhel mezi 0 a 360 stupni, pak se posune dopředu

```
REPEAT 10 [
  RIGHT RANDOM 360
  FORWARD 50
]
```

Tento kód způsobí, že želva vytvoří náhodný obrazec, protože pokaždé zvolí jiný úhel otočení.

Příklad: Náhodné barvy a tvary

- Náhodný vstup lze kombinovat s různými vlastnostmi, například s barvami nebo velikostí obrazců.
- To vede k velmi kreativním a unikátním výsledkům.

Příklad: Náhodná barva pera a kreslení čtverců různých velikostí

```
REPEAT 5 [
  SETPC RANDOM 16  ; Nastaví náhodnou barvu pera
  FORWARD RANDOM 100
  RIGHT 90
]
```

V tomto příkladu želva kreslí čtverce různých velikostí a barev, což dává vizuálně zajímavý výsledek.

14. Podmínky v Logo

- **Podmínky** umožňují provádět příkazy pouze za určitých podmínek
- Klíčové slovo `IF` kontroluje, zda je podmínka splněna
- **Příklad podmínky**: Pokud je hodnota větší než 50, vykresli čáru

```
IF :size > 50 [FORWARD 100]
```

- Pomocí podmínek můžeme kód učinit interaktivním a reagovat na různé vstupy

15. Smyčky: REPEAT

- **REPEAT**: Opakuje blok kódu několikrát

```
REPEAT 5 [FORWARD 50 RIGHT 72]
```

- Vhodné pro jednoduché opakování, když víme přesný počet opakování

16. Smyčky: FOR

- **FOR**: Umožňuje iteraci s určitou hodnotou (např. procházení sekvencí)

```
FOR [i 1 5] [FORWARD :i * 10 RIGHT 36]
FOR [i 1 50 1] [FORWARD :i * 10 RIGHT 36]
```

- Umožňuje iterovat proměnné a provádět složitější výpočty

17. Příklad: Procházení seznamu pomocí FOR

- **Procházení seznamu**: Pomocí smyčky `FOR` můžeme iterovat přes jednotlivé prvky seznamu

```
MAKE "mylist [10 20 30 40]
FOR [i 1 COUNT :mylist] [
  SHOW ITEM :i :mylist
]
```

- Tento příklad zobrazí všechny prvky seznamu `mylist` jeden po druhém

18. Smyčky: WHILE

- **WHILE**: Opakuje blok příkazů, dokud je podmínka pravdivá

```
MAKE "count 0
WHILE [count < 5] [
  FORWARD 50
  RIGHT 72
  MAKE "count :count + 1
]
```

- Vhodné pro dynamické situace, kdy počet opakování závisí na podmínce

19. Význam smyček v algoritmickém myšlení

- Smyčky jsou základní stavební kámen algoritmického myšlení
- Umožňují automatizovat opakované úlohy
- Pomáhají nám řešit problémy efektivněji a s menším množstvím kódu

20. Rekurze v Logo

- **Rekurze** je technika, kdy procedura volá sama sebe
- Umožňuje řešit složité problémy dělením na menší podproblémy
- Vhodná pro vytváření fraktálů a opakujících se vzorů

21. Příklad rekurze: Jednoduchá rekurzivní procedura

- **Příklad**: Nakreslení rekurzivního obrazce

```
TO recurse :size
  IF :size < 10 [STOP]
  FORWARD :size
  RIGHT 45
  recurse :size * 0.8
END

recurse 100
```

- Procedura `recurse` volá sama sebe s menší hodnotou, dokud není splněna podmínka

22. Význam rekurze v algoritmickém myšlení

- Rekurze pomáhá rozdělit složité problémy na jednodušší části
- Využívá se při generování fraktálů, hledání cest a dalších algoritmech
- Je klíčová pro pochopení principů, jako je děl a impera (rozděl a panuj)

23. Co je fraktál?

- **Fraktál** je geometrický útvar, který se opakuje na různých úrovních zvětšení
- Fraktály jsou často tvořeny pomocí rekurzivních algoritmů
- Příklady fraktálů: Sierpinského trojúhelník, Kochova vločka

24. Fraktály v programování Logo

- Logo umožňuje snadno vytvářet fraktály pomocí rekurze
- Fraktály jsou skvělý způsob, jak se naučit rekurzivní myšlení a vytvářet složité obrazce

25. Příklad: Kochova vločka

- **Kochova vločka** je jeden z nejjednodušších fraktálů
- Definována opakovaným dělením čáry na menší segmenty

```
TO koch :length
  IF :length < 5 [FORWARD :length STOP]
  koch :length / 3
  LEFT 60
  koch :length / 3
  RIGHT 120
  koch :length / 3
  LEFT 60
  koch :length / 3
END

TO snowflake :size
  REPEAT 3 [koch :size RIGHT 120]
END

snowflake 200
```

- Tento program kreslí Kochovu vločku o dané velikosti

26. Co je Kochova vločka?

- **Kochova vločka** je fraktál, který vzniká rekurzivním dělením čáry na menší části
- Začíná jako rovnostranný trojúhelník, kde každá strana je dále dělena do tvaru „vločky“
- Výsledný tvar má nekonečně složitý okraj, přičemž plocha je konečná
- Tento fraktál je ideální pro demonstraci základních principů rekurze a nekonečné detailnosti

27. Příklad: Sierpinského trojúhelník

- **Sierpinského trojúhelník** je další příklad fraktálu, který lze snadno vytvořit pomocí rekurze
- Definován dělením trojúhelníku na menší podtrojúhelníky

```
TO sierpinski :length
  IF :length < 10 [STOP]
  REPEAT 3 [
    FORWARD :length
    sierpinski :length / 2
    RIGHT 120
  ]
END

sierpinski 200
```

- Tento program kreslí Sierpinského trojúhelník rekurzivním způsobem, kdy každý trojúhelník se dále rozděluje na menší části

28. Co je Sierpinského trojúhelník?

- **Sierpinského trojúhelník** je jeden z nejznámějších fraktálů
- Vzniká rozdělením rovnostranného trojúhelníku na menší rovnostranné trojúhelníky
- Proces dělení pokračuje do nekonečna, čímž vytváří složitý obrazec s nekonečným množstvím prázdných míst
- Tento fraktál je příkladem samopodobnosti – menší části obrazce jsou podobné celku

29. Příklad: Pythagorův strom

- **Pythagorův strom** je fraktál složený z řady pravoúhlých trojúhelníků
- Definován postupným přidáváním menších trojúhelníků na základnu každého předchozího trojúhelníku

```
TO pythagoras :length :depth
  IF :depth = 0 [STOP]
  FORWARD :length
  LEFT 45
  pythagoras :length * 0.707 :depth - 1
  RIGHT 90
  pythagoras :length * 0.707 :depth - 1
  LEFT 45
  BACK :length
END

pythagoras 100 5
```

- Tento program kreslí Pythagorův strom pomocí rekurze, přičemž každý nový trojúhelník je menší než předchozí

30. Co je Pythagorův strom?

- **Pythagorův strom** je fraktál složený z pravoúhlých trojúhelníků, které se iterativně přidávají
- Začíná s jedním základním čtvercem, který je rozdělen na dvě části, a každá část tvoří nový čtverec
- Tento proces pokračuje, dokud nevznikne složitý, stromovitý tvar
- Pythagorův strom je skvělým příkladem rekurze a samopodobnosti a často se používá k demonstraci geometrických vlastností fraktálů

22. Shrnutí lekce

- Naučili jsme se základní stavební kameny jazyka Logo
  - Pohybové příkazy, procedury, proměnné, podmínky, smyčky, rekurze, seznam
- Ukázaly jsme si co fraktály

Děkuji za pozornost!