11. Řetězce, práce se soubory

Speciální znaky v řetězcích: `\"` `\n` `\t` `\\`

Délka řetězce: `StringLength[řetězec]`

Spojení řetězců: `StringJoin[řetězec`$_1$`,...,řetězec`$_n$`]`

Vložení řetězce na zadanou pozici: `StringInsert[původní řetězec,vkládaný řetězec,pozice]`

Výběr části řetězce: `StringTake[řetězec,{od,do}]`

Odstranění části řetězce: `StringDrop[řetězec,{od,do}]`

Obrácení řetězce: `StringReverse[řetězec]`

Hledání v řetězci: `StringPosition[řetězec,hledaný řetězec]`

Počet výskytů: `StringCount[řetězec,hledaný řetězec]`

Nahrazení: `StringReplace[řetězec,pravidlo]` `StringReplace[řetězec,seznam pravidel]`

Pomocí volby `IgnoreCase->True` lze zabránit rozlišování velkých a malých písmen.

Převod na velká/malá písmena: `ToUpperCase[řetězec]` `ToLowerCase[řetězec]`

Převod řetězce na seznam znaků: `Characters[řetězec]`

Převod řetězce na výraz: `ToExpression[řetězec]`

Převod výrazu na řetězec: `ToString[výraz]`

Posloupnost znaků v abecedě: `Alphabet[]` `Alphabet["jazyk"]`

Práce se soubory:

`Import["název souboru nebo URL"]` `Export["název souboru",exportovaný objekt]`

Podporované formáty lze zjistit pomocí `$ImportFormats` a `$ExportFormats`, další podrobnosti jsou v dokumentaci (*Listing of All Formats*).

Cvičení

1. Naprogramujte funkci pro šifrování řetězců pomocí Caesarovy šifry. Vstupem funkce je řetězec a přirozené číslo n. Každé písmeno řetězce bude nahrazeno písmenem, které vznikne cyklickým posunem v abecedě o n znaků doprava. (Např. pro $n = 1$ dojde k nahrazení `a->b`, `b->c`, ... , `z->a`.) Předpokládejte, že text neobsahuje háčky a čárky; můžete se omezit pouze na malá písmena anglické abecedy. Naprogramujte také inverzní funkci pro dešifrování textu. Najděte anglické slovo, jehož zašifrovaná podoba je `egdbdueq`.

Návod: Zkonstruujte pomocí `Alphabet[]` seznam přepisovacích pravidel a použijte `StringReplace`.

2. Cykloida je dráha pevně zvoleného bodu na obvodu kružnice, která se valí po přímce. Uvažujme jednotkovou kružnici, jejíž střed se v čase $t \geq 0$ nachází v bodě $(t, 1)$ (tj. kružnice se valí po přímce $y = 0$). Bod kružnice, který měl na počátku souřadnice $(0, 0)$, je v čase t na pozici $(t - \sin t, 1 - \cos t)$. Vyrobte animaci ve formátu GIF, která bude zobrazovat valící se kružnici s bodem, který za sebou zanechává stopu ve tvaru cykloidy. Omezte se např. na časový interval $t \in [0, 8\pi]$.

3. Součástí Mathematiky je slovník obsahující více než 90 000 anglických slov; jejich seznam lze získat pomocí `DictionaryLookup[]`.

a) Sestavte tabulku četností délek slov (pro každé n zjistěte, kolik je slov délky n), zobrazte výsledek graficky pomocí sloupcového diagramu. Jaká je průměrná délka slova ve slovníku? Jaké je nejdelší slovo?

b) Najděte všechna slova, která obsahují aspoň devět samohlásek.

c) Najděte nejdelší slovo, které lze sestavit ze sirek, tj. z písmen A, E, F, H, I, K, L, M, N, T, V, W, X, Y, Z. Nerozlišujte malá a velká písmena.

d) Které je nejčastější počáteční písmeno? Nerozlišujte malá a velká písmena.

Tip: Při řešení úloh se vám může hodit funkce `Tally` (nalezení četností prvků v seznamu) a predikát `SubsetQ` (test, zda je seznam podmnožinou jiného seznamu).

4. Následující funkce je inspirována jazykem LOGO, ve kterém lze zadávat instrukce pro pohyb pomyslné želvy a sledovat její dráhu:

```
turtle[s_String,angle_,startdir_]:=
Module[{dir=startdir,pos={0,0},path={{0,0}},left,right,i,ch=Characters[s]},
left=RotationMatrix[angle];right=RotationMatrix[-angle];
Do[Which[ch[[i]]=="+",dir=left.dir,ch[[i]]=="-",dir=right.dir,
ch[[i]]=="F",pos+=dir;AppendTo[path,pos]],{i,1,Length[ch]}];
ListLinePlot[path,Axes->False,AspectRatio->Automatic]]
```

Tato funkce očekává na vstupu řetězec `s` obsahující znaky `+`, `-`, `F`, dále úhel `angle` a počáteční směr `startdir`. Počáteční poloha je v bodě (0,0). Cyklus `Do` postupně prochází znaky řetězce `s`: Znak `F` znamená vykreslení úsečky, znaky `+`, `-` odpovídají otočení o úhel `angle` doleva nebo doprava.

a) Jaký útvar vykreslí `turtle["F++F++F",60 Degree,{1,0}]`?

b) Definujme `genString[s_String,rule_,i_]:=Nest[StringReplace[#,rule]&,s,i]`. Jaký obrázek vyrobí `turtle[genString["F++F++F","F"->"F-F++F-F",1],60.0 Degree,{1,0}]`? Zkuste změnit třetí argument `genString` na 2, 3, 4, atd.

c) Modifikujte funkci `turtle` tak, aby vstupní řetězec mohl obsahovat také znaky `[` a `]`. Kdykoliv načteme znak `[`, uložíme aktuální stav (tj. pozici a směr) na zásobník. Při načtení znaku `]` vyzvedneme ze zásobníku poslední uložený stav, přesuneme se na odpovídající pozici (bez toho, abychom kreslili čáru) a nastavíme správný směr.

Návod: Operace se zásobníkem lze implementovat pomocí seznamu: Stavy ukládejte vždy na začátek seznamu, vyzvedávejte je také ze začátku seznamu. Funkci `ListLinePlot` nepředávejte jeden seznam, ale seznam seznamů: Aktuální seznam ukončete vždy při načtení znaku `]`, zamezíte tím kreslení úsečky při přesunu na zapamatovanou pozici.

d) Vyzkoušejte `turtle[genString["F","F"->"F[+F]F[-F]F",5],180.0 Degree/7,{0,1}]`, `turtle[genString["F","F"->"FF+[+F-F-F]-[-F+F+F]",4],180.0 Degree/8,{0,1}]`.

Poznámka: O želví grafice a L-systémech se lze dočíst na `https://en.wikipedia.org/wiki/L-system`.