8. Anonymní funkce, funkcionální programování

Některé užitečné funkce pro práci se seznamy:

`First[seznam]` vrací první prvek zadaného seznamu.
`Last[seznam]` vrací poslední prvek zadaného seznamu.
`Rest[seznam]` vrací zadaný seznam bez prvního prvku.
`Most[seznam]` vrací zadaný seznam bez posledního prvku.
`Flatten[seznam]` odstraní všechny vnitřní složené závorky v zadaném seznamu.

Anonymní funkce jedné proměnné:	Anonymní funkce více proměnných:
`Function[proměnná,předpis]`	`Function[seznam proměnných,předpis]`
`Function[předpis s proměnnou #]`	`Function[předpis s proměnnými #1,#2,...]`
`(předpis s proměnnou #)&`	`(předpis s proměnnými #1,#2,...)&`

Funkcionální programování:

`Map[f,{`$\mathtt{x}_1$`,`$\mathtt{x}_2$`,...}]` spočte $\{f(x_1), f(x_2), \ldots\}$ (ekvivalentní zápis: `f/@{`$\mathtt{x}_1$`,`$\mathtt{x}_2$`,...}`).
`MapThread[f,{{`$\mathtt{x}_1$`,`$\mathtt{x}_2$`,...},{`$\mathtt{y}_1$`,`$\mathtt{y}_2$`,...},...}]` spočte $\{f(x_1, y_1, \ldots), f(x_2, y_2, \ldots), \ldots\}$.

`Nest` a `NestList` pracují s funkcí f jedné proměnné. Označme $f^n = f \circ \cdots \circ f$ (n-krát).
`Nest[f,x,n]` spočte $f^n(x)$.
`NestList[f,x,n]` spočte $\{x, f(x), f^2(x), \ldots, f^n(x)\}$.

`Fold` a `FoldList` pracují s funkcí f dvou proměnných.
`FoldList[f,x,{`$\mathtt{a}_1$`,`$\mathtt{a}_2$`,...}]` spočte $\{x, f(x, a_1), f(f(x, a_1), a_2), \ldots\}$.
`Fold[f,x,{`$\mathtt{a}_1$`,`$\mathtt{a}_2$`,...}]` vrací poslední prvek seznamu získaného z `FoldList`.

Cvičení

Při řešení následujících úloh nepoužívejte `Table` ani přístup k prvkům seznamu pomocí `[[...]]`.

1. Naprogramujte funkci, která dostane seznam $\{s_1, \ldots, s_n\}$, kde s_i jsou neprázdné seznamy, a vrátí seznam $\{x_1, \ldots, x_n\}$, kde x_i je náhodně vybraný prvek seznamu s_i. (Použijte `RandomChoice` a `Map`.)

2. Víte, co dělá funkce definovaná předpisem `f[s_List]:=Map[{#,Count[s,#]}&,Union[s]]`?

Poznámka: Stejného výsledku lze dosáhnout pomocí zabudované funkce `Tally`.

3. Naprogramujte pomocí `MapThread` funkci, která dostane jako argumenty dva stejně dlouhé seznamy $\{a_1, \ldots, a_n\}$ a $\{b_1, \ldots, b_n\}$, a vrátí seznam hodnot `True` nebo `False` odpovídající tomu, zda jsou prvky na odpovídajících místech seznamů stejné. Např. pro seznamy `{1,2,3,4,5}` a `{1,0,3,0,5}` bude výsledkem `{True,False,True,False,True}`.

4. Jestliže `A` je libovolná matice, co je `MapThread[List,A]`?

5. Posloupnost $\{V_n\}_{n=1}^{\infty}$ je definována rekurentně: $V_1 = \sqrt{\frac{1}{2}}$, $V_n = \sqrt{\frac{1}{2} + \frac{1}{2}V_{n-1}}$. Naprogramujte funkci pro výpočet V_n pomocí `Nest`. Tato posloupnost se objevuje ve Viétově vzorci pro výpočet π:
$\pi = \lim_{n\to\infty} a_n$, kde $a_n = 2/\prod_{k=1}^{n} V_k$. Spočítejte a_{20} a porovnejte, na kolik míst se shoduje s π.

6. V úloze o zajatcích stojí n osob v kruhu. Každá druhá je popravena; toto se opakuje tak dlouho, dokud nezůstane poslední živý zajatec, kterému je udělena milost. Naším úkolem je naprogramovat funkci, která dostane seznam osob tak, jak jsou seřazeny v kruhu, a určí zajatce, jenž přežije. Můžeme postupovat např. takto:

```
zajatci[s_List]:=Nest[Rest[RotateLeft[#]]&,s,Length[s]-1]
```

a) Co se stane, jestliže místo `Nest` použijeme `NestList`?

b) Jak by se dalo dosáhnout toho, abychom dostali seznam zajatců v pořadí, ve kterém jsou popravováni? (Zkuste využít předchozí část.)

c) Upravte uvedenou funkci tak, aby místo každého druhého zajatce byl popraven každý k-tý.

7. Označme symbolem $A(z, n)$ množinu všech komplexních čísel tvaru $a_1 z + a_2 z^2 + \cdots + a_n z^n$, kde $a_i \in \{0, 1\}$.

a) Naprogramujte funkci, která vygeneruje tuto množinu.

Návod: Každou přípustnou n-tici $\{a_1, \ldots, a_n\}$ lze chápat jako zápis čísla z množiny $\{0, \ldots, 2^n - 1\}$ ve dvojkové soustavě; všechny takové n-tice lze získat pomocí `IntegerDigits[Range[0,2^n-1],2,n]`. Hodnotu $a_1 z + a_2 z^2 + \cdots + a_n z^n$ pak dostaneme jako skalární součin $\{a_1, \ldots, a_n\}$ s $\{z^1, \ldots, z^n\}$.

b) Naprogramujte funkci, která zobrazí množinu $A(z, n)$. Převeďte komplexní čísla na dvojice reálných čísel a použijte funkci `ListPlot` s volbou `AspectRatio->Automatic`. Zobrazte množiny $A(0.5+0.5i, 15)$, $A(0.65 - 0.3i, 14)$, $A(0.2 + 0.6i, 14)$, $A(0.8 + 0.2i, 15)$.

8. Naprogramujte pomocí `Fold` funkci pro výpočet faktoriálu.

9. Víte, co dělá funkce definovaná předpisem `f[s_List]:=Fold[(10*#1+#2)&,0,s]`? Vstupem je libovolně dlouhý seznam čísel z množiny $\{0, \ldots, 9\}$.

10. Definujte pomocí `Fold` funkci

$$f(n,x) = \sqrt{1 + 2\sqrt{1 + 3\sqrt{1 + 4 \cdots + n\sqrt{1+x}}}}, \quad n \in \mathbf{N},\ x > -1$$

(pro $n = 1$ je $f(1, x) = \sqrt{1+x}$).

11. Permutaci π na množině $\{1, \ldots, n\}$ lze zadat pomocí seznamu hodnot $\{\pi(1), \ldots, \pi(n)\}$, nebo také pomocí rozkladu na nezávislé cykly; např. permutaci zadané seznamem $\{2, 5, 4, 3, 1\}$ odpovídá rozklad $\{\{1, 2, 5\}, \{3, 4\}\}$. Naprogramujte funkci, která dostane rozklad permutace na cykly (v uvedeném tvaru), a převede jej na seznam hodnot $\{\pi(1), \ldots, \pi(n)\}$.

Návod: Převeďte každý cyklus na seznam pravidel, tj. např. $\{1, 2, 5\}$ na $\{1 \to 2,\ 2 \to 5,\ 5 \to 1\}$; můžete k tomu použít např. konstrukci `MapThread[Rule,{{x1,...,xn},{y1,...,yn}}]` (vzpomeňte si, že vnitřní reprezentace `x->y` je `Rule[x,y]`). Pravidla získaná ze všech cyklů aplikujte na seznam $\{1, \ldots, n\}$.