10. Procedurální programování

Přiřazení hodnoty proměnné: `proměnná=hodnota`
Modifikace hodnoty proměnné: `x+=c  x-=c  x*=c  x/=c  x++  x--`

Složený příkaz: `příkaz`$_1$`;` $\cdots$ `;příkaz`$_n$

Cykly:
`Do[příkaz,{i,min,max}]  Do[příkaz,{i,min,max,krok}]`
`Do[příkaz,{i,min`$_1$`,max`$_1$`},{j,min`$_2$`,max`$_2$`},...]`
`While[podmínka,příkaz]`
`For[inicializace,podmínka,inkrementace,příkaz]`

Modul s lokálními proměnnými: `Module[seznam lokálních proměnných,příkaz]`

Podmíněné příkazy:
`If[podmínka,příkaz]  If[podmínka,příkaz,alternativní příkaz]`
`Which[podmínka`$_1$`,příkaz`$_1$`,...,podmínka`$_n$`,příkaz`$_n$`]`

Vstup a výstup:
`Print[výraz`$_1$`,...,výraz`$_n$`]`
`Input[]  Input[výzva]`
`InputString[]  InputString[výzva]`

Cvičení

1. Dirichletova věta říká, že pro každou dvojici nesoudělných přirozených čísel a, d existuje nekonečně mnoho prvočísel p takových, že $p \bmod d = a$. (Ekvivalentně: Aritmetická posloupnost $\{a + nd\}_{n=0}^{\infty}$ obsahuje nekonečně mnoho prvočísel.) Naprogramujte funkci, která pro zadané hodnoty a, d, k vrátí seznam prvních k prvočísel s uvedenou vlastností. Vyzkoušejte např. pro $a = 5$, $d = 6$, $k = 15$.

2. Pro libovolné přirozené číslo n označme jako $\Pi(n)$ součin všech cifer čísla n, např. $\Pi(77) = 7 \times 7 = 49$. Pro pevně zvolené n definujme posloupnost danou rekurentním vzorcem $c_{i+1} = \Pi(c_i)$ a počátečním členem $c_1 = n$. Multiplikativní vytrvalost čísla n je pak nejmenší přirozené číslo k takové, že $c_k = c_{k+1}$. Např. pro $n = 77$ platí

$$\{c_i\}_{i=1}^{\infty} = (77, 49, 36, 18, 8, 8, \ldots),$$

tedy multiplikativní vytrvalost čísla 77 je 5. Naprogramujte funkci, která pro danou hodnotu m najde přirozené číslo $n \in \{1, \ldots, m\}$, které má největší multiplikativní vytrvalost (pokud jich je více, najděte první z nich). Použijte tuto funkci pro $m = 10^6$ a pro nalezené číslo n s největší multiplikativní vytrvalostí zkonstruujte příslušný seznam $\{c_1, \ldots, c_k\}$.

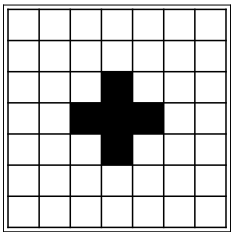
Návod: K získání seznamu cifer daného čísla použijte `IntegerDigits`.

3. „Hra života“ (*game of life*), jejímž autorem je matematik John Conway, se odehrává v síti $n \times n$ políček, kterou lze reprezentovat pomocí matice. Každé políčko je buď prázdné (0), nebo obsahuje živý organismus (1). Hra probíhá v tzv. generacích: První generace odpovídá počátečnímu stavu (tj. matici, kterou zadal uživatel). Je-li v k-té generaci na pozici (i, j) živý organismus, pak přežije do $(k + 1)$-ní generace právě tehdy, jestliže má 2 nebo 3 živé sousedy (jinak umírá na osamělost nebo přelidnění). Je-li v k-té generaci pozice (i, j) prázdná, pak se zde v $(k + 1)$-ní generaci narodí nový organismus, má-li prázdné políčko právě 3 živé sousedy. Za sousední se v obou případech považují políčka, která mají společnou hranu nebo roh (tj. každé políčko má aspoň 3 a nejvýše 8 sousedů).

a) Naprogramujte funkci, která pro zadanou matici nul a jedniček určí, jak bude vypadat následující generace (výstupem je tedy opět matice).

Návod: Pomocí funkce `ArrayPad` orámujte zadanou matici nulami. Při určování počtu živých sousedů políčka (i, j) pak není nutné rozlišovat pozice na okrajích matice od ostatních. K určení počtu živých sousedů lze pomocí `Sum` sečíst hodnoty ve vhodné podmatici 3×3 a poté odečíst hodnotu uprostřed.

b) Pomocí předchozí části naprogramujte funkci `life[m,n]`, která vrátí seznam s obrázky prvních n generací určených počáteční maticí m. Pro zobrazení matic použijte `ArrayPlot` s volbami `FrameTicks->True, Mesh->True, Frame->True`. Vygenerujte pomocí funkce `life` prvních 9 generací počínaje následující (černá políčka jsou živé organismy):



Zobrazte výsledek ve formě animace pomocí `ListAnimate[life[m,n]]`.

c) Vezměte jako počáteční stav náhodnou matici o rozměrech 50×50 a vyrobte animaci znázorňující prvních 500 generací.