

10. PROCEDURÁLNÍ PROGRAMOVÁNÍ

Přiřazení hodnoty proměnné: `proměnná=hodnota`

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

Složený příkaz: `příkaz1; ... ;příkazn`

Cykly:

`Do[příkaz, {i, min, max}]` `Do[příkaz, {i, min, max, krok}]`

`Do[příkaz, {i, min1, max1}, {j, min2, max2}, ...]`

`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[podmínka1, příkaz1, ..., podmínkan, příkazn]`

Vstup a výstup:

`Print[výraz1, ..., výrazn]`

`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 najde prvních k prvočísel s uvedenou vlastností. Vyzkoušejte např. pro $a = 5, d = 6, k = 15$.

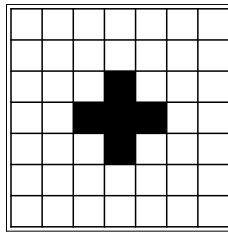
Návod: `NextPrime[n]` vrací nejmenší prvočíslo větší než n .

2. „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` orámuje 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):



Tip: `ListAnimate[life[m,n]]` vyrobí ze seznamu obrázků animaci.

c) Umíte najít nějaký počáteční stav ve čtverci o rozměrech aspoň 3×3 , při kterém organismy nikdy nevyumřou?

3. Permutaci π na množině $\{1, \dots, n\}$ lze zadat pomocí seznamu hodnot $\{\pi(1), \dots, \pi(n)\}$, 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 permutaci ve tvaru seznamu a rozloží ji na nezávislé cykly.

Návod: Použijte pomocný seznam délky n , který na začátku naplníte hodnotami `False`. Kdykoliv během výpočtu zařadíte číslo i do některého cyklu, zapíšete `True` na i -tou pozici pomocného seznamu.