PROG²

Zadanie 2

> **Nezabudnite zadanie nahrať do AIS** do príslušného miesta odovzdania (odovzdajte len zdrojový `.c` súbor).

Dôležité informácie

- **Termín odovzdania:** Odovzdajte do AIS (22. marec 2020, 23:59), osobné prezentovanie zadaní sa určí neskôr
- **Počet bodov:** 8
- **Prezentácia Z2**

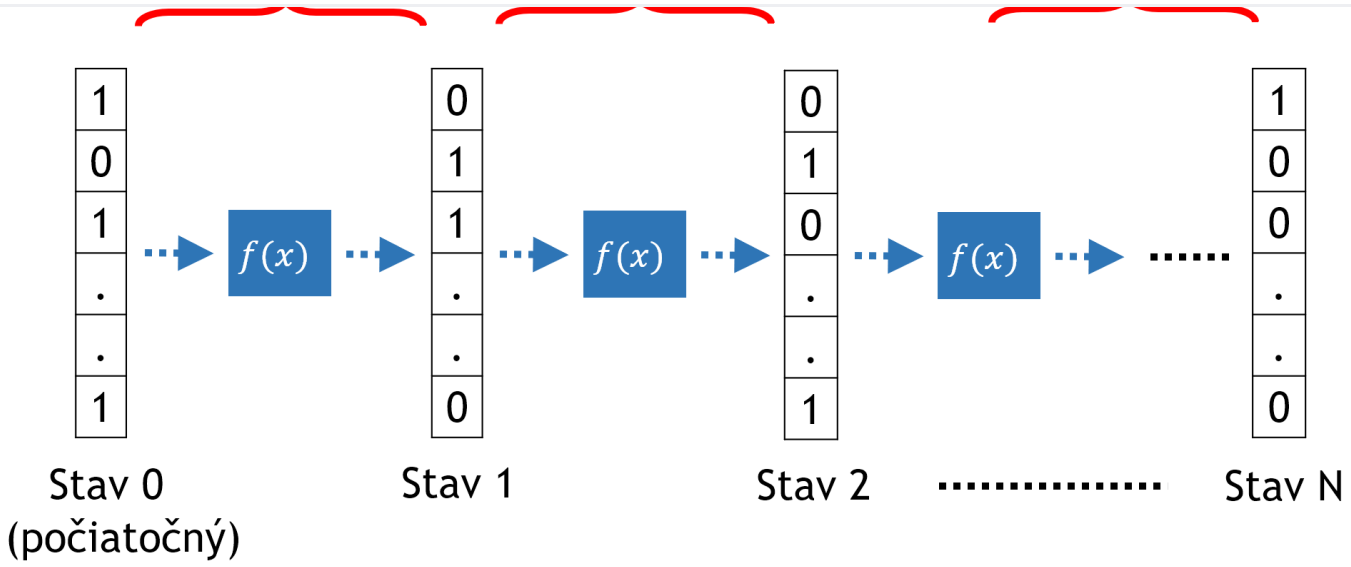
Krátky popis

Cieľom zadania č. 2 je naprogramovať v jazyku C konzolovú aplikáciu, ktorá bude simulovať prácu **stochastického celulárneho automatu** (jedná sa o špecifický druh **celulárneho automatu**). Celulárny automat je systém, pomocou ktorého vieme simulovať rôzne javy. Celulárny automat je reprezentovaný svojim aktuálnym stavom. Stav automatu je 1D pole hodnôt (v tomto zadaní pracujeme s 1D automatom, dajte si pozor, v literatúre sa dosť často uvádza 2D automat), ktoré je vyplnené hodnotami 0 alebo 1. Automat sa dokáže vyvíjať v čase, t.j. prechádzať do nových stavov podľa určitých pravidiel. Prechod do nového stavu je riadený tzv. prechodovou funkciou. Prechodová funkcia vypočíta konkrétnu hodnotu nového stavu na základe zodpovedajúcej hodnoty v pôvodnom stave a jej okolia (susedné hodnoty v poli). Prechodová funkcia nebude v prípade tohto zadania deterministická (nebude pevne stanovené pravidlo na prechod zo starého stavu do nového), ale namiesto toho bude používať pravdepodobnostné pravidlá, t.j. automat sa do konkrétneho nového stavu dostane s určitou pravdepodobnosťou.

Pre lepšie pochopenie si pozrite **prezentáciu so zadaním č.2**.

Obr. 1 Znázornenie prechodu automatu do nového stavu pomocou prechodovej funkcie (okolie bunky v starom stave je vstupom do prechodovej funkcie, ktorá následne s využitím pravdepodobnostných pravidiel vypočíta hodnotu v novom stave). Prechodová funkcia, ktorú treba použiť v tomto zadaní bude vysvetlená nižšie.

PROG²



Obr. 2 Znázornenie vývoja automatu v čase (viaceré iterácie). Prechodová funkcia je znázornená modrou farbou. Počet iterácií si zvolí používateľ na začiatku programu.

Hlavné body zadania:

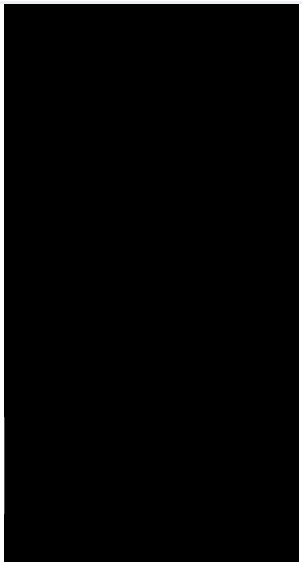
- napísať funkciu na inicializáciu počiatočného stavu automatu
- napísať funkciu na zadanie pravidiel pre výpočet nového stavu automatu
- napísať funkciu výpočet nového stavu automatu (ideálne vytvoriť aj pomocné podfunkcie)
- napísať funkciu na vizualizáciu stavov automatu do konzoly (je nutné oddeliť zdrojový kód logiky automatu od jeho vizualizácie v konzole)

V skratke, vašou úlohou je naprogramovať stochastický celulárny automat podľa uvedených pokynov a vizualizovať jeho vývoj v čase, t.j. postupnosť jeho stavov (výpisom do konzoly).

Ukážka (video)

Video ukážka zachytávajúca beh programu v konzole a interakciu s používateľom. Najprv treba zadať počet iterácií automatu. Následne používateľ zadá počet pravidiel a načíta ich z klávesnice do poľa. Potom nasleduje výpis vývoja automatu (počiatočný stav nasledovaný postupným prechádzaním do nových stavov). Táto ukážka slúži ako inšpirácia.

PROG²



0:00

Ďalšie video ukážky (použité sú iné parametre automatu):

- **Ukážka 1**: používateľ zadá pravidlá s nízkymi pravdepodobnosťami vygenerovania jednotky (hviezdičky sú vypisované nariedko)
- **Ukážka 2**: používateľ zadá pravidlá s vysokými pravdepodobnosťami vygenerovania jednotky (hviezdičky sú vypisované nahusto)
- **Ukážka 3**: používateľ zadá pravidlá tak, aby sa vždy vygenerovala jednotka (nové stavy automatu sú vyplnené len jednotkami)
- **Ukážka 4**: iná dĺžka stavu (dĺžka stavu je 10)
- **Ukážka 5**: simulácia šírenia epidémie

Postup

V tejto sekcii zhrniem všetky základné operácie súvisiace so stochastickým celulárnym automatom, ktoré je potrebné vo vašom zadaní naprogramovať (každú operáciu treba implementovať ako samostatnú funkciu).

1. Inicializácia počiatočného stavu automatu

Toto zadanie je určené na precvičenie práce s poliami. Stav automatu je reprezentovaný ako 1D pole. Na začiatku programu treba inicializovať počiatočný stav automatu, t.j. vyplniť pole náhodne hodnotami 0 alebo 1. Treba to urobiť pomocou vlastnej funkcie, ktorá dostane na vstup stav (1D pole) a jeho platnú dĺžku (celé číslo). Pri každom spustení vášho programu by mali byť hodnoty počiatočného stavu náhodné (t.j. nezabudnúť použiť knižničnú funkciu `srand`).

PROG²

Obr. 3 Znázornenie počiatočného stavu (pole s názvom `state`) a jeho vyplnenie náhodnými bitmi 0/1. Vlastné makro `STATE_MAX_LEN` určuje kapacitu poľa.

2. Určenie pravidiel na výpočet nového stavu automatu

Jedná sa o pravdepodobnostné pravidlá, ktoré budú použité prechodovou funkciou pri výpočte nového stavu. Rovnako ako stav automatu, aj pravidlá budú reprezentované ako 1D pole. Hodnoty tohto poľa sú celé čísla v intervale `<0,100>`, t.j. pravdepodobnosti vyjadrené v percentách. Pravidlá sa určia na začiatku programu. Treba si napísať vlastnú funkciu, ktorá nastaví pravidlá. Vstupom do tejto funkcie bude pole s pravidlami a jeho dĺžka. Používateľ zvolí koľko pravidiel chce zadať, avšak tento počet nesmie presahovať dĺžku poľa. Ak používateľ zadá menší počet pravidiel ako je dĺžka poľa, tento počet je potrebné vrátiť z funkcie, aby zvyšok programu poznal reálny počet pravidiel.

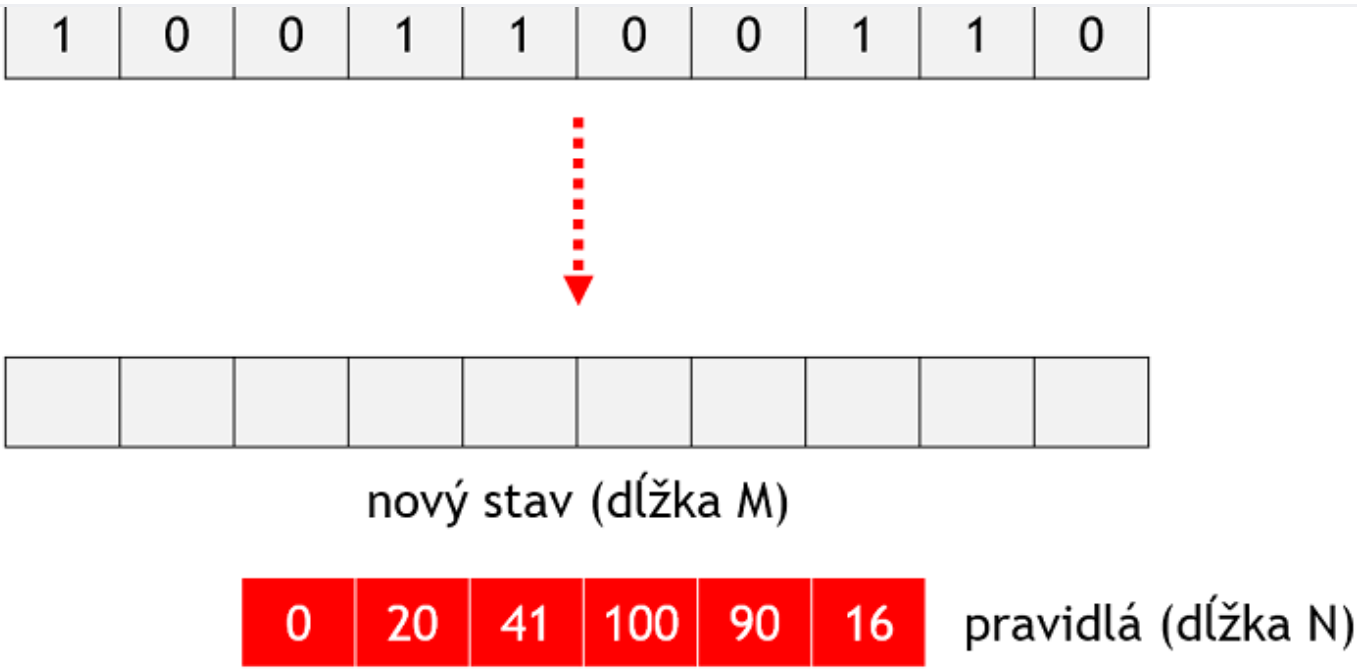
Pravidlá budú určené jedným z nasledujúcich spôsobov (výber je na vás):

- používateľ ich manuálne zadá z klávesnice (načíta čísla z intervalu `<0,100>` do poľa)
- získajú sa zo súboru pomocou presmerovania štandardného vstupu (použije sa funkcia `scanf` ako aj v prvej variante, presmerovanie bolo ukázané na seminári č. 3).

Obr. 4 Znázornenie príkladu pravidiel (pole s názvom `rules`) a interpretácia významu ich hodnôt. Vlastné makro `RULES_MAX_LEN` určuje kapacitu poľa.

3. Výpočet nového stavu automatu

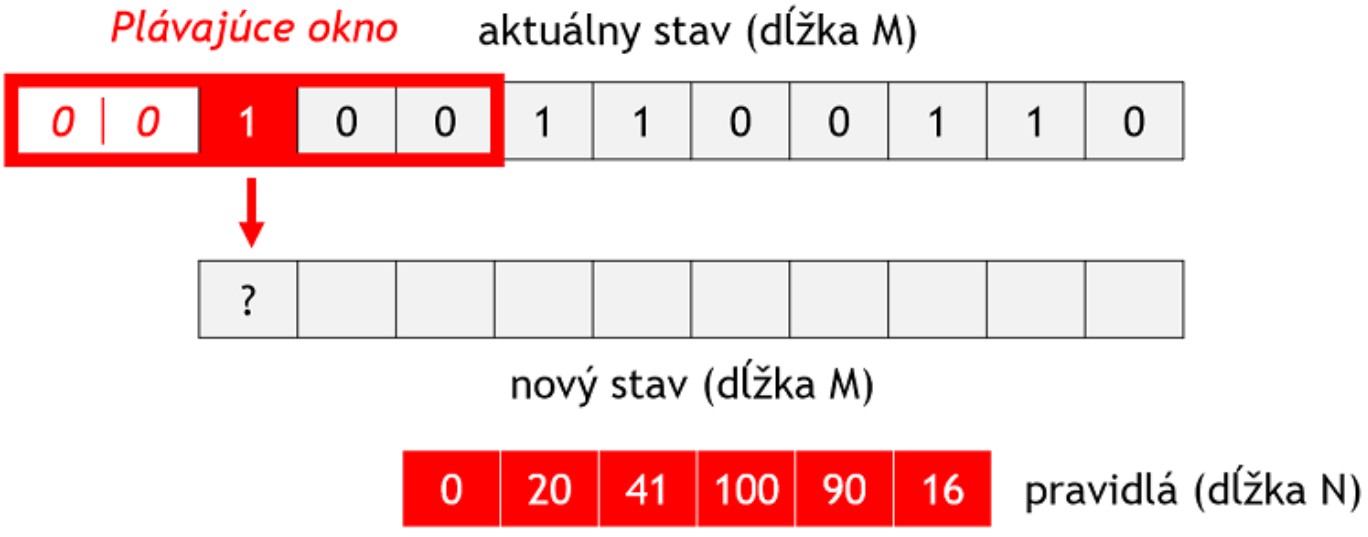
Pred samotnou simuláciou vývoja automatu, používateľ zadá počet iterácií, t.j. koľko krát prejde automat do nového stavu. Následne sa v cykle začnú počítať nové stavy automatu. Na výpočet nového stavu automatu si napíšte vlastnú funkciu, v ktorej budete v cykle opakovane volať prechodovú funkciu. Vstupom do tejto funkcie bude starý a nový stav, ich dĺžka, pole s pravidlami a jeho dĺžka.



Obr. 5 Znázornenie situácie pred prechodom automatu do nového stavu

Prechodová funkcia

Prechodová funkcia pri výpočte nového stavu bude používať tzv. plávajúce okno. Plávajúce okno predstavuje okolie zvolenej bunky aktuálneho stavu, ktoré sa použije na výpočet bunky v novom stave. Šírka plávajúceho okna musí byť nepárne číslo (aby sa dal jasne indexom určiť stred okna). Ak používateľ zadal na začiatku programu `N` pravidiel, potom šírka tohto okna musí byť `N-1` (hodnota `N-1` musí byť nepárna).



Obr. 6 Znázornenie plávajúceho okna s dĺžkou `N-1`.

Prechodová funkcia vypočíta `i`-tu hodnotu v novom stave nasledovne:

1. Spočítajú sa jednotky v plávajúcom okne (napíšte si na to samostatnú funkciu). Plávajúce okno má stred v indexe `i`. Počet jednotiek v okne označme ako `K`. Ak časť okna presahuje za okraj poľa, v tom prípade treba za chýbajúce hodnoty v okne dosadiť hodnotu 0.

PROG²

zodpovedajúcu pozíciu. Na generovanie 0/1 s uvedenými pravdepodobnosťami si napíšte zvlášť funkciu (parametrom bude pravdepodobnosť `p` a funkcia vráti 0 alebo 1).

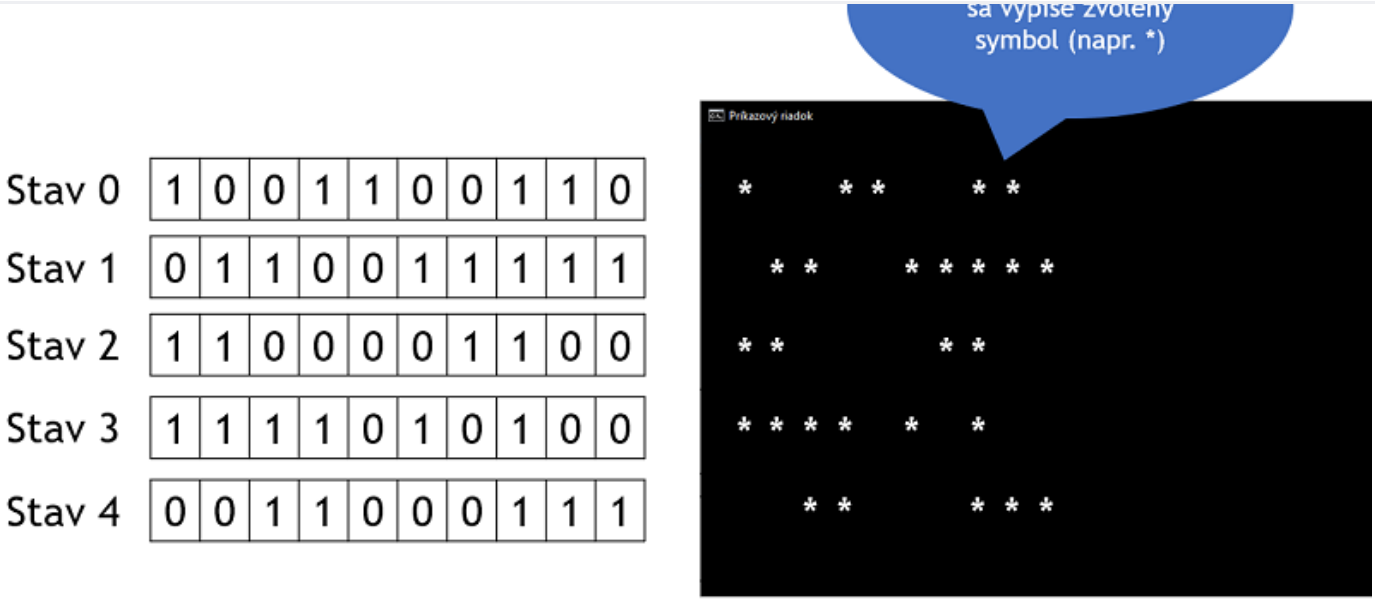
3. Okno sa posunie o jednu pozíciu doprava a celý proces sa opakuje od bodu 1. Postupným posúvaním okna vypočítame všetky hodnoty v novom stave automatu.

Obr. 7 Znázornenie procesu výpočtu hodnoty nového stavu pomocou plávajúceho okna a poľa s pravdepodobnostnými pravidlami.

4. Vizualizácia stavov automatu

Vývoj automatu v čase (postupnosť jeho stavov) treba vhodne vizualizovať v konzolovom okne. Treba si napísať vlastnú funkciu, ktorá bude mať na vstupe pole so stavom automatu a jeho dĺžkou. Funkcia vykreslí stav do jedného riadku. Hodnota 1 bude zobrazená v konzole pomocou vami zvoleného znaku, napr. `*`. Hodnota 0 bude vypísaná ako medzera. Počet stavov automatu, ktoré sa vykreslia závisí od počtu iterácií, ktoré zadá používateľ na začiatku programu.

PROG²



Obr. 8 Výpis postupnosti stavov automatu do konzoly (príklad).

Kostra riešenia a potrebné funkcie

Toto je navrhovaná kostra riešenia, ktorá slúži ako inšpirácia. Jedná sa o hrubý náčrt, takže kód je len na ukážku, nie je kompilovateľný. Nezabudnite, že sa hodnotí aj kvalita vášho programátorského návrhu a rozdeľte celý program do funkcií. Vyvarujte sa na maximum používaniu globálnych premenných.

```c
#define _CRT_SECURE_NO_WARNINGS
#define STATE_MAX_LEN 80 // zadajte tak aby to sedelo so sirkou konzoly
#define RULES_MAX_LEN 10 // toto cislo je na vas

#include <stdio.h>
#include <stdlib.h>
#include <time.h>

// Funkcia na inicializaciu pociatocneho stavu automatu
// Pozn. vstupne pole sa nahodne vyplni hodnotami 0 alebo 1
void init_state(/* pole so stavom */, /* dlzka pola */) {
    // todo
    // vo funkcii sa pomocou cyklu for prejde cele vstupne pole
    // a na jednotlive pozicie sa priradi nahodna hodnota 0/1
}

// Funkcia na nastavenie pravdepodobnostnych pravidiel pre
// potreby generovania noveho stavu automatu
// Pozn. ak pouzivatel zada mensi pocet pravidiel ako je dlzka
// pola, tento novy pocet funkcia vrati.
int init_rules(/* pole s pravidlami */, /* dlzka pola */) {
    // todo
    // 1. funkcia vyzve pouzivatela na zadanie poctu pravidiel (zada N pravidiel)
    // 2. v cykle sa nacita do pola N pravidiel
```

PROG²

```
28
29      // funkcia nakoniec vrati realne zadany pocet pravidiel (t.j. N)
30  }
31
32  // Funkcia na generovanie 0 alebo 1, kde vstupna pravdepodobnost p
33  // predstavuje pravdepodobnost vygenerovania 1 (0 sa vygeneruje s
34  // pravdepodobnostou 100-p)
35  int generateBit(int p) {
36      // todo
37      // funkcia vrati vygenerovany bit s tym, ze s pravdepodobnostou p
38      // to bude jednotka
39
40      // priklady:
41      // * ak p = 20, mame 20% sancu vygenerovania jednotky
42      // * ak p = 100, vzdy sa vygeneruje jednotka
43      // * ak p = 0, vzdy sa vygeneruje nula
44  }
45
46  // Funkcia, ktora spocita vo vstupnom poli (okne) jednotky
47  int sumInWindow(/* vstupne pole */, /* dlzka pola */) {
48      // todo
49  }
50
51  // Funkcia, ktora do jedneho riadku vytlaci stav automatu
52  void printState(/* pole so stavom */, /* dlzka pola */) {
53      // todo
54  }
55
56  // Funkcia na vypocet noveho stavu pomocou povodneho stavu a pola s pravidlami
57  // Vstupy:
58  //    * old_state - pole s povodnym stavom
59  //    * new_state - pole s novym stavom
60  //    * m - dlzka pola so stavom (plati pre povodny aj novy stav)
61  //    * rules - pole s pravidlami
62  //    * n - dlzka pola s pravidlami
63  void generateNextState(/* old_state */, /* new_state */, /* m */, /* rules */, /* n *
64      // todo
65      // funkcia bude pomocou plavajuceho okna a pola s pravidlami
66      // pocitat hodnoty noveho stavu
67
68      // pozn. nezabudnite tu vyuzit vase funkcie 'sumInWindow' a 'generateBit'
69
70      // pozn. funkcia nic nevracia
71  }
72
73  // hlavna funkcia na spustenie automatu, pricom pocet iteracii je 'num'
74  void cellular_automaton(int num) {
75      // todo
76
77      // tu sa vytvoria vsetky potrebne polia a premenne
78      // polia so stavmi budu mat dlzku 'STATE_MAX_LEN'
79      // pole s pravidlami bude mat dlzku 'RULES_MAX_LEN'
```

PROG²

```
83      // 3. v cykle sa 'num'-krat vypocita (zavola funkcia generateNextState)
84      // a vypise novy stav automatu (zavola funkcia printState)
85  }
86
87  // testovanie
88  int main()
89  {
90      // pouzivatel zada pocet iteracii automatu a zavola hlavnu spustaciu funkciu
91      cellular_automaton(/* zadany pocet iteracii */);
92      return 0;
93  }
```

Bodovanie

Body sú rozdelené nasledovne:

- **[1b]** Funkcia na inicializáciu počiatočného stavu automatu
- **[1b]** Funkcia na zadanie/nastavenie pravidiel výpočtu nového stavu
- **[3b]** Funkcia na výpočet nového stavu (použije sa prechodová funkcia, ktorá využíva zadané pravidlá)
- **[1b]** Funkcia na vizualizáciu stavu automatu v konzole
- **[2b]** Práca s poľom (ohodnotenie toho ako kvalitne váš program pracuje s poliami, ako využívate polia vo funkciách a pod.)

Extra body

Za extra snahu a námahu môžu byť udelené extra body po zvážení cvičiacim. Extra funkcionalita môže byť napr.:

- môžete implementovať 2D automat (napr. **Game of Life automat**)
- pokročilá vizualizácia výpočtu nového stavu (napr. znázornenie posúvania plávajúceho okna v poli počas generovania nového stavu)

Zdroje

- **Stochastic cellular automaton**
- **Cellular automaton**
- **Celulární automat**
- **1D celulárny automat**
- **Elementary cellular automaton (Wolfram)**
- **1D Cellular Automata and the Edge of Chaos**

PROG²