PROG²

Zadanie 5

(1. skúškové zadanie)

> **Nezabudnite zadanie nahrať do AIS** do príslušného miesta odovzdania (odovzdajte len príslušný zdrojový `.c` súbor). Súbor je následne nutné v AIS potvrdiť tlačidlom niekoľkokrát za sebou až pokiaľ neuvidíte oznam o úspešnom odovzdaní.

Dôležité informácie

- **Termín odovzdania:** Odovzdajte do AIS, **17. máj 2020, 23:59**
- **Počet bodov:** 10
- Odovzdávané súbory a formát ich pomenovania (ID nahraďte vašim AIS ID číslom):
  - `ID.main.z5.c` (celá implementácia zadania spolu s `main` funkciou)
- Podmienkou získania bodov za zadanie je, aby bol váš kód kompilovateľný prekladačom `gcc` s prepínačom `-Wall` bez chýb a upozornení (teda 0 errors, 0 warnings). Na otestovanie tejto nutnej podmienky máte rôzne možnosti:
  - Použite Linux (`gcc` je jeho bežnou súčasťou)
  - Vo Windows nainštalujte prostredie **Cygwin** alebo ľubovoľné IDE prepojené s **MinGW** a tam pomocou príkazu `gcc` skompilujte svoj kód. **Návod na základné použitie gcc**.
- Nezabudnite dodržať termín odovzdania do AIS, **oneskorené odovzdanie nebude akceptované**.

Obsah

- **Príprava**
- **Cieľ zadania**
- **Postup**
  - **1. Príkaz na spustenie programu**
  - **2. Stav CGoL**
  - **3. Prechodová funkcia CGoL**
  - **4. Formát výstupu do konzoly, s prepínačom -v**
  - **5. Formát vstupov a výstupov, bez prepínača -b**
  - **6. Formát vstupov a výstupov, s prepínačom -b**
  - **7. Odporúčania k implementácii**

PROG²

- Testovanie a bodovanie
- Konzultovanie zadania

Príprava

- Zopakujte si vybrané témy z jazyka C:
  - Polia: (**C Arrays**, **Multidimensional Arrays**)
  - Reťazce: (**string.h**)
  - Dynamická pamäť: **malloc**, **calloc**, **realloc**, **free**
  - Štruktúry: (**C struct**)
  - Súbory (aj binárne): **fopen**, **fclose**, **fprintf**, **fscanf**, **fseek**, **ftell**, **rewind**, **fread**, **fwrite**
    - Práca s binárnymi súbormi: **Binary I/O Functions**
  - Bitové operácie: **Bitwise operators in C**
- Ďalšie online zdroje
  - **Learn C Programming (programiz.com)**
  - **Free Interactive C Tutorial (learn-c.org)**
- Tí, ktorí pracujete vo Visual Studio nezabudnite na:
  - makro `_CRT_SECURE_NO_WARNINGS` povolí používanie funkcií ako `scanf` a pod. (treba ho napísať úplne na začiatok programu pred miesto vloženia ostatných `.h` súborov)
  - makro `_USE_MATH_DEFINES` povolí používanie matematických konštánt z `math.h`

Cieľ zadania

Naprogramujte v jazyku C známy 2D celulárny automat s názvom **Conway's Game of Life** (ďalej len CGoL). Výsledkom vašej práce bude terminálová/konzolová aplikácia, ktorej správanie bude riadené zadanými prepínačmi a argumentami.

Postup

1. Príkaz na spustenie programu

Štruktúra príkazu, ktorým spustíte váš program:

```
./z5 [-v] [-b] R S T input_file output_file
```

Popis jednotlivých častí príkazu:

- `-v` Špecifikuje, že každá iterácia CGoL bude vypísaná do terminálu/konzoly (viď **časť 4**).

PROG²

- `S` Počet stĺpcov stavu (torusu). Musí platiť `S>0`.
- `T` Počet iterácií CGoL, ktoré váš program vykoná. Musí platiť `T>=0`.
- `input_file` Názov vstupného súboru, v ktorom sa nachádza počiatočný stav CGoL.
- `output_file` Názov výstupného súboru, do ktorého sa zapíše konečný stav CGoL.

Program vykoná `T` iterácií CGoL nad stavom (torusom) s veľkosťou `R` riadkov a `S` stĺpcov. Na načítanie počiatočného stavu/zápis konečného stavu automatu sa použije:

- textový súbor `input_file` / `output_file` (štandardný režim fungovania)
- binárny súbor `input_file` / `output_file` (len ak je zadaný voliteľný prepínač `-b`)

Časti príkazu, ktoré sú v hranatých zátvorkách sú **voliteľné**, t.j. program funguje s nimi aj bez nich. Ostatné časti príkazu sú **povinné**. Ak nie je zadaný povinný prepínač alebo je v nesprávnom tvare (alebo jeho parametre), tak program skončí bez výstupu.

2. Stav CGoL

Stav automatu je uložený v 2D poli binárnych hodnôt (`0` = mŕtva bunka, `1` = živá bunka). Každá bunka má **8 susedov** (2 susedia horizontálne, 2 susedia vertikálne a 4 susedia diagonálne). Stav automatu je reprezentovaný ako **torus** s dynamickou veľkosťou `R` x `S` (zadané pri spustení).

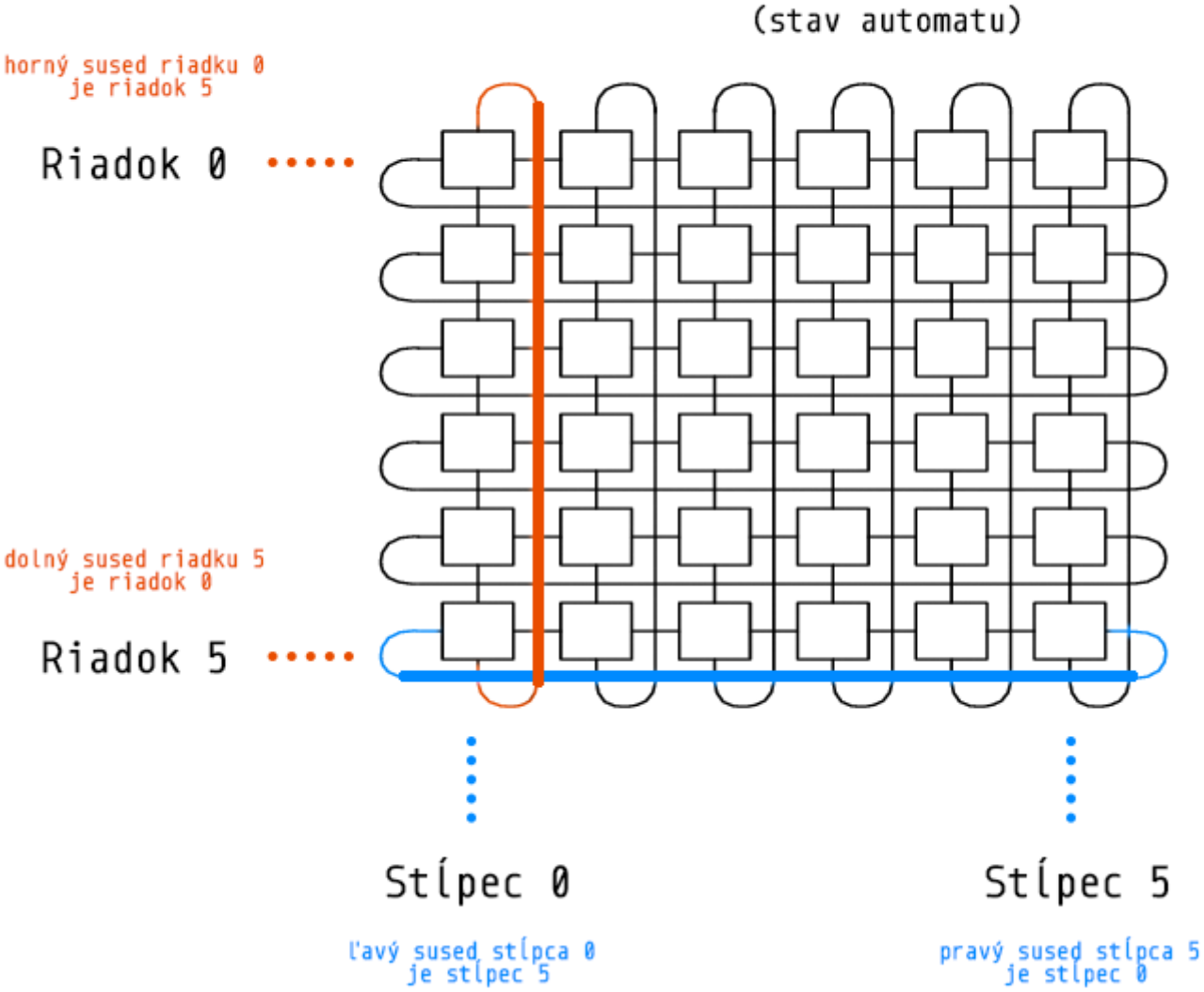
Torus

Torus znamená, že:

- ľavým susedom stĺpca `0` je stĺpec `S-1`
- pravým susedom stĺpca `S-1` je stĺpec `0`
- horným susedom riadku `0` je riadok `R-1`
- dolným susedom riadku `R-1` je riadok `0`

HINT: počítajte indexy riadkov a stĺpcov modulo `R`, `S`, pozor na záporné čísla.

PROG²



Obr.1 Ukážka torusu s rozmermi 6x6

3. Prechodová funkcia CGoL

Prechodová funkcia CGoL je definovaná nasledovne:

1. Každá živá bunka s menej než dvoma živými susedmi zomrie.
2. Každá živá bunka s dvoma alebo tromi živými susedmi zostáva žiť.
3. Každá živá bunka s viac než tromi živými susedmi zomrie.
4. Každá mŕtva bunka s práve tromi živými susedmi ožije.

4. Formát výstupu do konzoly, s prepínačom -v

Ak používateľ zadá pri spustení prepínač `-v`, výstupy všetkých iterácií (vrátane počiatočného a konečného stavu) CGoL budú navyše vypísané do terminálu/konzoly. Výpis stavu do konzoly má nasledovný formát.

- počet riadkov stavu = `R`
- počet stĺpcov stavu = `S`
- obsahom stavu sú živé (znak hviezdičky `*`) a mŕtve bunky (medzera).

PROG²

5. Formát vstupov a výstupov, bez prepínača -b

Na I/O operácie sa použijú **textové** súbory.

Vstupný súbor

Ak sa pri spustení nezadá prepínač `-b`, tak sa použije vstupný textový súbor `input_file` obsahujúci počiatočný stav CGoL vyjadrený znakmi `'0'` / `'1'`. Hodnoty '0' a '1' v textovom súbore sa chápu ako jedna postupnosť (bez ohľadu na ostatné formátovacie znaky). Z textového súboru sa načíta maximálne `R` x `S` hodnôt (od začiatku súboru). Pole obsahujúce počiatočný stav sa plní číslami 0 a 1 po riadkoch využitím znakov `'0'` / `'1'` z textového súboru. Všetky ostatné znaky v textovom súbore sa ignorujú (preskakujú pri načítaní).

- ak je v textovom súbore **viac** ako `R` x `S` znakov `'0'` alebo `'1'`: **zvyšné znaky sa ignorujú**
- ak je v textovom súbore **menej** ako `R` x `S` znakov `'0'` alebo `'1'`: **pole sa doplní nulami**

Výstupný súbor

Do výstupného textového súboru sa zapíše konečný stav CGoL (stav po poslednej iterácii). Na zápis konečného stavu CGoL sa použijú znaky `'0'` / `'1'`. Zápis prebieha po riadkoch. Každý riadok končí znakom `'\n'`.

6. Formát vstupov a výstupov, s prepínačom -b

Na I/O operácie sa použijú **binárne** súbory. Ak sa použije prepínač `-b`, v prípade, že hodnota `S` nie je deliteľná 8, zaokrúhli sa na najbližší násobok 8 nahor (napr. 21 sa zaokrúhli na 24).

Vstupný súbor

Ak sa pri spustení zadá prepínač `-b`, tak sa použije vstupný binárny súbor `input_file` obsahujúci binárne kódované hodnoty počiatočného stavu CGoL. Z binárneho súboru sa načíta maximálne `R` x `S` /8 bajtov. Počiatočný stav sa plní po riadkoch po osmiciach stĺpcov, pričom každá osmica sa nastaví na hodnoty podľa aktuálne načítaného bajtu. Každý bajt kóduje postupnosť binárnych číslic stavu od least significant bit (LSB). Napr. načítaný bajt `'A'`, s ASCII hexadecimálnym kódom `0x41` (dekadicky 65) kóduje postupnosť 8 bitov `10000010`. Každý bajt je platný. Ak postupnosť bajtov vo vstupnom súbore skončí skôr ako sa naplní celý počiatočný stav, v tom prípade sa stav doplní nulami.

Výstupný súbor

Do výstupného binárneho súboru sa zapíše konečný stav CGoL (stav po poslednej iterácii). Výstupný binárny súbor sa vytvára ekvivalentne k vstupnému. Osmice hodnôt v riadku sa kódujú naspäť do binárneho čísla (bajtu) od LSB, a vytvorené bajty sa zapisujú do výstupného súboru.

PROG²

7. Odporúčania k implementácii

Je vhodné používať 2D dynamické polia, ale môžete použiť aj 1D pole a prepočítavať súradnice. V každom prípade odporúčam vytvoriť si štruktúru na reprezentáciu stavu CGoL, ktorá bude obsahovať dáta a rozmery.

Odporúčané funkcie

`read_text_file()`

- Vstup: súbor/meno súboru, rozmery
- Výstup: načítaný stav automatu

`read_binary_file()`

- Vstup: súbor/meno súboru, rozmery
- Výstup: načítaný stav automatu

Výsledný načítaný stav po oboch funkciách (`read_text_file` , `read_binary_file`) by mal byť v rovnakom tvare.

`next_state()`

- Vstup: stav automatu
- Výstup: nasledujúci stav automatu

`evolve()`

- Vstup: stav, počet iterácií
- Výstup: finálny stav

`write_text_file()`

- Vstup: súbor/meno súboru, stav
- Výstup: nie je, zapíše zadaný stav automatu do súboru v textovom móde

`write_binary_file()`

- Vstup: súbor/meno súboru, stav

PROG²

Samozrejme, môžete si vytvoriť aj ďalšie pomocné funkcie (napr. dekódovanie/zakódovanie bajtu).

8. Odporúčané testovanie vášho programu

- Overte si, že po 0 iteráciách dostanete rovnaký výstup ako bol vstup (pri textovom súbore až na formátovacie znaky, resp. záverečné nuly).
- Vytvorte si pomocný testovací program, ktorý vám konvertuje binárne súbory na textové a naopak. Overte si, či dávajú rovnaké výsledky.
- Pomocou nasledovnej postupnosti príkazov si vyskúšajte, že súbory `2a.txt` a `2b.txt` sú rovnaké (za `R` a `S` dosaďte vlastné kladné hodnoty):

```
./z5 R S 2 0.txt 2a.txt
./z5 R S 1 0.txt 1.txt
./z5 R S 1 1.txt 2b.txt
```

- Pomocou nasledovnej postupnosti príkazov si vyskúšajte, že súbory `2a.bin` a `2b.bin` sú rovnaké (za `R` a `S` dosaďte vlastné kladné hodnoty):

```
./z5 -b R S 2 0.bin 2a.bin
./z5 -b R S 1 0.bin 1.bin
./z5 -b R S 1 1.bin 2b.bin
```

- Skontrolujte si, že výstupy vášho programu korešpondujú napr. s **Toad vzorom** (rozmer stavu 6x8). Perióda opakovania tohto vzoru je 2 (viď animácia pod textom).

`toad0.txt` (počiatočný stav) (`toad0.bin` obsahuje tieto hexa zápisy bajtov: `0x00 0x08 0x12 0x12 0x04 0x00`)

```
00000000
00010000
01001000
01001000
00100000
00000000
```

`toad1.txt` (stav po 1. iterácii) (`toad1.bin` obsahuje tieto hexa zápisy bajtov: `0x00 0x00 0x1c 0x0e 0x00 0x00`)

```
00000000
00000000
00111000
01110000
00000000
00000000
```

PROG²

Na stiahnutie

- 0.txt
- 0.bin
- toad0.txt
- toad1.txt
- toad0.bin
- toad1.bin

9. Ukončenie programu

Program/aplikácia skončí bez výstupu:

- keď nie je zadaný povinný prepínač/argument programu
- keď je prepínač v nesprávnom tvare (alebo jeho parametre)

Testovanie a bodovanie

Zadanie bude hodnotené pomocou automatizovaných testov (otestuje sa séria rôznych prepínačov a argumentov s rôznymi hodnotami) a preto je potrebné, aby váš program vypisoval na `stdout` /zapisoval do súboru len požadovaný výsledok - žiadne iné znaky, slová, pomocné hlásenia, dekoračné výpisy a pod. (toto bola dosť častá chyba študentov pri zadaní 3 a 4). Ak túto podmienku váš program nesplní, je dosť možné, že za zadanie nezískate žiadne body. Preto výrazne odporúčam dôkladne si kontrolovať vaše výstupy/výpisy.

Body sú orientačne rozdelené nasledovne:

- **[6b]** Za správnu funkcionalitu vášho programu pri spracovaní textových súborov
- **[4b]** Za správnu funkcionalitu vášho programu pri spracovaní binárnych súborov

Poznámka: Nezabudnite, že nutnou podmienkou je, aby bol váš program kompilovateľný prekladačom `gcc` so zapnutým prepínačom `-Wall` bez chýb a upozornení (0 errors a 0 warnings). Vo vlastnom záujme si toto otestujte.

Konzultovanie zadania

Všetky vaše otázky smerujúce k zadaniu č.5 môžete zasielať na STUBA e-mailové adresy svojich cvičiacich alebo sa zapojiť do živej diskusie na našom Discord serveri (invite linka bola poslaná cez AIS) v kanáli `#zadanie5` .

Stay tuned.