Požiadavky na semestrálnu prácu z predmetu princípy operačných systémov

Všeobecné pravidlá

- Semestrálnu prácu vypracováva tím pozostávajúci z 1 alebo 2 študentov. **Tím pracuje na semestrálnej práci samostatne**, pričom po úspešnej obhajobe si členovia tímu sami určia, ako si rozdelia pridelené body.
- Každý tím musí odovzdať semestrálnu prácu prostredníctvom aktivity na Moodli.
- **Originalita každej semestrálnej práce bude kontrolovaná (aj s minuloročnými semestrálnymi prácami).**
- Pri vypracovaní semestrálnej práce je možné využiť **generatívnu umelú inteligenciu**, avšak výslednému riešeniu musí **detailne rozumieť každý člen tímu**,
- Semestrálnu prácu musí obhajovať celý tím. **Pri obhajobe môžu byť členovia tímu požiadaní o úpravu zdrojového kódu alebo doplnenie ďalšej funkcionality. Ak toto nezvládnu v stanovenom čase, za semestrálnu prácu dostávajú 0 bodov.**

Všeobecné požiadavky

Cieľom semestrálnej práce je vytvoriť **viacprocesovú viacvláknovú** aplikáciu, ktorá využíva prostriedky **medziprocesnej komunikácie** (IPC). Semestrálna práca musí spĺňať nasledujúce požiadavky:

- implementácia v jazyku C,
- návrh v súlade s **princípmi objektového programovanie** (hlavne zapuzdrenie a modularita),
- **žiadne úniky pamäte** (toto je nutné explicitne ukázať s využitím nástrojov, ktoré sú na to určené),
- kód musí byť logicky rozčlenený do .h a .c súborov,
- semestrálna práca musí obsahovať **vlastný Makefile (alebo súbor CMakeLists.txt)**, pomocou ktorého bude možné zostaviť jednotlivé časti semestrálnej práce (napr. klient a server). Makefile musí obsahovať **minimálne nasledujúce ciele**:
  - server,
  - client,
  - all,
  - clean,
- **pri vývoji musí byť využitý verzionovací systém** (preferovane **GitLab**):
  - v rámci verzionovacieho systému musia byť dokladované aspoň **2 príspevky od každého člena tímu** a musí byť vykonaný aspoň **1 merge vetví projektu**,
  - pred obhajobou semestrálnej práce je nutné vyučujúcemu, u ktorého budete prácu obhajovať, udeliť prístupové právo **Reporter**,
- so semestrálnou prácou odovzdávate:
  - programátorskú dokumentáciu (podľa zverejnenej šablóny), v ktorej opíšete:
    - štruktúru projektu (s využitím UML),
    - procesy, z ktorých pozostáva aplikácia,
    - použitie a účel vlákien,
    - typy a účel použitých prostriedkov medziprocesnej komunikácie;
    - synchronizačné problémy, ktoré ste museli riešiť, a prostriedky použité na ich riešenie,
    - ďalšie kľúčové problémy, ktoré ste v rámci semestrálnej práce riešili,
    - informáciu kde a ako ste použili **generatívnu umelú inteligenciu**. Ak ste ju nepoužili, je nutné explicitne napísať, že semestrálne práca bola vypracovaná bez použitia generatívnej umelej inteligencie. **Za použitie generatívnej umelej inteligencie sa pokladá aj jej využitie pri tvorbe dokumentácie alebo ostatných súborov potrebných na kompiláciu a spustenie aplikácie.**
  - používateľskú dokumentáciu, v ktorej opíšete spustenie a ovládanie jednotlivých častí aplikácie,
- celá semestrálna práca bude zbalená do jedného .zip súboru, ktorý odovzdáte prostredníctvom aktivity na Moodli. V .zip súbore budú **iba**:
  - .h súbory,
  - .c súbory,
  - Makefile alebo súbor CMakeLists.txt,
  - ostatné súbory **nevyhnutné na kompiláciu a spustenie aplikácie** (textové, grafické a pod.),
  - programátorská a používateľská dokumentácia.

Námety na semestrálnu prácu

Hadík pre jedného alebo viacerých hráčov – klasická hra určená pre jedného alebo viacerých hráčov (https://en.wikipedia.org/wiki/Snake_(video_game_genre)). **Aplikácia má spĺňať nasledujúce funkčné požiadavky:**

1. ľubovoľný hráč môže vytvoriť hru pre jedného alebo viacerých hráčov;
2. v hre sa generuje toľko ovocia, koľko hadíkov sa nachádza v hre; jednotlivé kusy ovocia musia byť vygenerované na rôznych políčkach, ktoré môžu byť navštívené každým hadíkom;
3. hra bude mať dva herné režimy:
   - štandardný – hra končí, ak v priebehu 10 sekúnd, odkedy z nej zmizol posledný hadík, sa do nej nepripojí žiaden hráč;
   - časový – hra končí, keď vyprší čas určený pre hru. Čas sa definuje na začiatku hry;
4. hra bude ponúkať dva typy herných svetov:
   - herný svet bez prekážok – v hernom svete sú iba hadíci a ovocie; ak sa hadík ocitne na jednom konci herného sveta, zobrazí sa na opačnom konci;
   - herný svet s prekážkami – okrem hadíkov a ovocia sa v hernom svete vyskytujú aj prekážky. Herný svet môže byť definovaný súborom, alebo môže byť generovaný náhodne. Pri náhodnom generovaní je nutné zabezpečiť, aby každé pole herného sveta, na ktorom sa nenachádza prekážka, bolo dosiahnuteľné hadíkom;
5. hra končí pre konkrétneho hráča, ak jeho hadík narazí sám do seba, do prekážky alebo do iného hadíka, alebo vyprší čas určený pre hranie hry (pozri bod 3). Ak hra skončí iba pre konkrétneho hráča, tento hráč sa môže do nej opäť pripojiť s novým hadíkom. Ak hra skončí pre všetkých hráčov, pre každého hadíka sa zobrazia nahraté body a jeho celkový čas v hre. Po skončení hry sa do nej nemôže pripojiť žiaden hráč;
6. hráč vytvárajúci hru definuje:
   - rozmery herného sveta (s výnimkou prípadu, ak je herný svet načítaný zo súboru);
   - v prípade časového herného režimu (pozri bod 3) aj čas určený pre hru;
7. v rámci hry sa musí zobrazovať počet nahratých bodov každého hráča a celkový čas behu hry;
8. ľubovoľný hráč môže v ľubovoľnom okamžiku pozastaviť svoju hru. Pozastavením sa hráč dostáva do hlavného menu. Hadík hráča, ktorý pozastavil hru, ostáva v hre, avšak nehýbe sa. Ak sa hráč vráti do hry, jeho hadík sa po 3 sekundách začne opäť hýbať. Ak hráč definitívne odíde z hry, jeho hadík zmizne;
9. hlavné menu má obsahovať nasledujúce položky:
   - nová hra – spustí sa nová hra. Pred samotným spustením si hráč zvolí herný režim a typ herného sveta. Ak hra umožňuje načítať herný svet zo súboru, hráč musí mať možnosť zvoliť si súbor, ktorý chce otvoriť pre načítanie herného sveta. Pri vytváraní novej hry sa určí, či je určená iba pre jedného alebo viacerých hráčov;
   - pripojenie k hre – hráč sa pripojí k existujúcej hre pre viacerých hráčov. Hráč sa môže pripojiť k hre v ľubovoľnom okamžiku, ak hra nebola ukončená. Hráč sa môže pripojiť aj k hre, z ktorej predtým odišiel. Po pripojení k hre sú pohyby všetkých hadíkov v hre prerušené na 3 sekundy;
   - pokračovanie v hre – možnosť je prístupná len, ak hráč pozastavil práve hranú hru. Umožňuje hráčovi pokračovať v pozastavenej hre. Ak si po pozastavení hry hráč zvolí možnosť nová hra, tak hadík hráča z hry zmizne;
   - koniec – korektné ukončenie aplikácie.

Náhodná pochôdzka – cieľom je vytvoriť aplikáciu, ktorá bude simulovať pohyb opilca (radšej budeme hovoriť o chodcovi, https://en.wikipedia.org/wiki/Random_walk) v dvojrozmernom svete pozostávajúcom z políčok. Chodec so môže v každom okamžiku posunúť o jedno políčko hore, dole, vľavo alebo vpravo. Smer posunu je daný pravdepodobnosťou, ktorá predstavuje vstup do simulácie. Aplikácia umožní vypočítať pre každé políčko dvojrozmerného sveta priemerný počet krokov potrebných na dosiahnutie stredu sveta (políčko [0, 0]), ako aj pravdepodobnosť, s akou chodec dosiahne stred sveta za predpokladu, že môže vykonať najviac K posunov. **Aplikácia má spĺňať nasledujúce funkčné požiadavky:**

1. ľubovoľný používateľ môže vytvoriť simuláciu prístupnú pre jedného alebo viacerých používateľov;
2. simulácia poskytuje dva módy:
   - interaktívny – počas simulácie sa vykresľuje aktuálna trajektória chodca;
   - sumárny – na jednotlivých políčkach sa vypisuje priemerný počet posunov potrebný na dosiahnutie políčka [0, 0] alebo pravdepodobnosť, s akou sa chodcovi podarí dostať z daného políčka na políčko [0, 0] za predpokladu, že môže vykonať najviac K posunov; hodnota K sa definuje pri spustení simulácie;
3. simulácia sa spúšťa nad jedným z nasledujúcich typov sveta:
   - svet bez prekážok – v simulovanom svete sa nachádza iba chodec; ak sa chodec ocitne na jednom konci herného sveta a v ďalšom posune by sa mal nachádzať mimo svet, objaví sa na opačnom konci sveta;
   - svet s prekážkami – okrem chodca sa v simulovanom svete vyskytujú aj prekážky. Prekážka sa nemôže vyskytovať na pozícii [0, 0]. Svet môže byť definovaný súborom, alebo môže byť generovaný náhodne. Pri náhodnom generovaní je nutné zabezpečiť, aby každé políčko herného sveta, na ktorom sa nenachádza prekážka, bolo dosiahnuteľné z pozície [0, 0];
4. simulácia skončí, ak nastane niektorý z nasledujúcich prípadov:
   - vykonali sa všetky replikácie;
   - používateľ, ktorý vytvoril simuláciu, sa rozhodol ju ukončiť;
5. po skončení simulácie sa jej výsledok uloží do súboru, ktorého názov špecifikoval používateľ počas vytvárania simulácie. Tento súbor môže byť neskôr použitý pre opätovné spustenie simulácie;
6. do simulácie sa môže pripojiť ľubovoľný používateľ. Ak v simulácii nie je žiaden používateľ, tak po pripojení do simulácie sa používateľovi zobrazí sumárny mód. Ľubovoľný používateľ pripojený do simulácie môže zmeniť mód, pričom tento sa zmení všetkým používateľom pripojeným do simulácie. Ak sa používateľom zobrazuje sumárny mód, tak jednotliví používatelia si môže individuálne prepínať typ zobrazenia – priemerný počet posunov potrebných na dosiahnutie políčka [0, 0] alebo pravdepodobnosť dosiahnutia políčka [0, 0] po maximálne K posunoch;
7. používateľ vytvárajúci novú simuláciu definuje:
   - rozmery sveta (s výnimkou prípadu, ak je svet načítaný zo súboru);
   - počet replikácií;
   - pravdepodobnosti, podľa ktorých sa chodec rozhoduje, na ktoré z okolitých štyroch políčok sa posunie (súčet týchto pravdepodobností musí byť rovný 1);
   - hodnotu K definujúcu maximálny počet krokov, ktoré môže chodec vykonať (pozri bod 2);
   - súbor, do ktorého bude uložený výsledok simulácie;
8. používateľ vytvárajúci simuláciu opätovným spustením už ukončenej simulácie definuje:
   - súbor, z ktorého bude simulácia načítaná;
   - počet replikácií;
   - súbor, do ktorého bude uložený výsledok simulácie;
9. pod jednou replikáciou sa rozumie postup, ktorý pre každé políčko, ktoré neobsahuje prekážku, vygeneruje náhodnú trajektóriu, ktorá začína na danom políčku a končí na políčku [0, 0];
10. v rámci simulácie sa musí zobrazovať číslo aktuálnej replikácie a celkový počet replikácií;
11. hlavné menu má obsahovať nasledujúce položky:
    - nová simulácia – spustí sa nová simulácia. Pred samotným spustením si používateľ zvolí typ simulovaného sveta. Ak aplikácia umožňuje načítať svet zo súboru, používateľ musí mať možnosť zvoliť si súbor, ktorý chce otvoriť pre načítanie sveta. Pri vytváraní novej simulácie sa definuje, či je určená pre jedného alebo viacerých používateľov;
    - pripojenie k simulácii – používateľ sa pripojí k existujúcej simulácii. Používateľ sa môže pripojiť k simulácii v ľubovoľnom okamžiku, ak simulácia ešte neskončila;
    - opätovné spustenie simulácie – obnoví sa simulácia, ktorej výsledok bol uložený do súboru. Pri obnovení simulácie sa definuje, či je určená pre jedného alebo viacerých používateľov;
    - koniec – korektné ukončenie aplikácie.

Požiadavky na architektúru a implementáciu semestrálnej práce

Z pohľadu architektúry a implementácie **musí aplikácia, ktorá je výstupom semestrálnej práce, spĺňať nasledujúce požiadavky. Nesplnenie ktorejkoľvek z požiadaviek P 1 – P 10 má za následok získanie 0 bodov zo semestrálnej práce.**

P 1. Aplikácia alebo jej časť musí byť spustiteľná na stroji **frios2.fri.uniza.sk**.
P 2. Pri spustení aplikácie vzniká proces **klient**.
P 3. Pri vytvorení novej hry/simulácie vzniká proces **server**.
P 4. Ak aplikácia podporuje viacero klientskych procesov, tak k procesu server vytvorenému jedným klientom sa môže pripojiť ľubovoľný počet iných klientov, pričom na pripojenie môžu použiť niektorý z nižšie uvedených prostriedkov IPC.
P 5. Proces server existuje, pokiaľ beží ním spravovaná hra/simulácia bez ohľadu na to, či proces klient, ktorý ho vytvoril, beží alebo už skončil. Ak hra/simulácia skončí, proces server zaniká.
P 6. Ak aplikácia podporuje viacero klientskych procesov, tak na jednom počítači môže bežať ľubovoľný počet procesov typu klient a ľubovoľný počet procesov typu server, pričom ľubovoľný klient sa môže pripojiť k ľubovoľnému serveru, ak pozná jeho IPC rozhranie.
P 7. Klient je zodpovedný za prijímanie vstupu od používateľa, prácu v menu, vykresľovanie herného/simulovaného sveta a prácu s perzistentnými údajmi (v prípade hry napr. uchovávanie súborov s vytvorenými mapami herných svetov).
P 8. Server je zodpovedný za hernú/simulačnú logiku (v prípade hry napr. vytvorenie nového herného sveta alebo jeho načítanie zo súboru, alebo jeho vygenerovanie, pohyby hráčov, generovanie ovocia, detekcia kolízií).
P 9. Klienti a server budú navzájom komunikovať prostredníctvom niektorého z nasledujúcich prostriedkov IPC:
    - zdieľaná pamäť,
    - dátovody,
    - sokety.
P 10. Ak aplikácia podporuje viaceré prostriedky IPC, tak IPC rozhrania, ktoré umožnia pripojiť sa ostatným klientom, sa definujú pri vytvorení procesu server.
P 11. Klient alebo server by mali rozumným spôsobom využívať vlákna (napr. pri hraní hry je vhodné, aby mal klient dve vlákna, z ktorých jedno bude zodpovedné za prijímanie vstupu od používateľa a jeho posielanie serveru a druhé bude zodpovedné za prijímanie údajov zo servera a vykresľovanie herného sveta).
P 12. Voliteľná požiadavka – **lokálny grafický klient** – proces klient, ktorý bude bežať na lokálnom počítači. Tento klient nedokáže vytvoriť nový proces server, ale dokáže sa pripojiť pomocou soketov len k existujúcemu procesu server bežiacemu na inom počítači v počítačovej sieti. Lokálny grafický klient musí mať grafické rozhranie a môže byť implementovaný v ľubovoľnom programovacom jazyku a na ľubovoľnom operačnom systéme.

Bodovanie semestrálnej práce

Maximálny počet bodov závisí od **univerzálnosti riešenia**. Pod univerzálnosťou sa rozumie množstvo klientov, ktorí sa dokážu pripojiť k procesu sever (iba 1 klient alebo N klientov, kde N > 2).

Hodnotená oblasť	Maximálny počet bodov	
	N klientov	1 klient
IPC – zdieľaná pamäť	6	3
IPC – dátovody	6	3
IPC – sokety	6	3
Súčasná podpora viacerých rozhraní IPC	3	-
Splnenie funkčných požiadaviek	30	15
Kvalita objektového návrhu s ohľadom na funkčné požiadavky	10	5
Správna synchronizácia a práca s vláknami s ohľadom na funkčné požiadavky	20	10
Implementácia klienta s ohľadom na funkčné požiadavky	5	5
Implementácia servera s ohľadom na funkčné požiadavky	10	5
Lokálny grafický klient	10	-
Používateľský zážitok	10	5
Makefile alebo CMakeLists.txt	2	2
Programátorská dokumentácia	4	4
Používateľská dokumentácia	2	2
Spolu	**124**	**62**