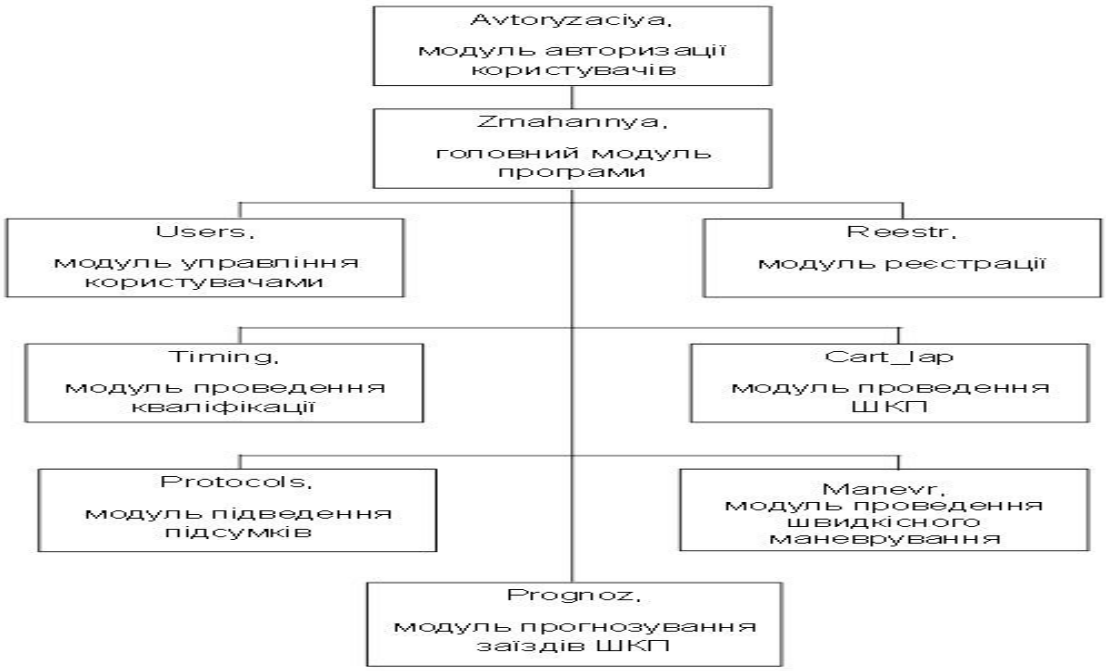

Рисунок 2 - Діаграма модулів програми

Модуль фігурного маневрування є додатковим модулем програми, оскільки дана дисципліна зустрічається на змаганнях з картингу тільки певної вікової категорії учасників, а саме до 22 років. Вихідні дані модуля не впливають на результати команд чи учасників в абсолютних заліках.

Висновок

Розглянуто алгоритм і структуру програми, а також діаграму прецедентів. Описано структуру модулів, які описують кожен етап проведення змагань. Розроблене програмне забезпеченняспрощує роботу суддів, автоматизує основні етапи роботи, прискорює обробку вхідних даних змагань та виконує їх збереження для зручного використання в майбутньому.

Список використаних джерел

1. Нечаєв В.П. Автоматизоване робоче місце менеджера / В.П. Нечаєв, Н.В. Гринь. - Кривий Ріг: «ІДА», 2009. - 146с.
2. Страхарчук А.Я. Інформаційні системи і технології в банках: Навч. посіб. Рекомендовано МОН / А.Я. Страхарчук, В.П. Страхарчук — К., 2010. — 515 с.

УДК 004.4

МЕТОДИКА ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ІНФОРМАЦІЙНОЇ СИСТЕМИ ПРИ ПЕРЕДАЧІ ВЕЛИКИХ ОБ'ЄМІВ ДАНИХ

Яркун В.І.
Національний університет «Львівська політехніка», студент

Вступ

Стрімкий розвиток інформаційних технологій дійшов до такого рівня, що дозволяє обробляти, передавати, використовувати великі порції даних, не затрачаючи часу на їх перевірку на правильність та цілісність. Ймовірною проблемою синхронізації кількох процесів при передачі даних може бути не ефективне використання ресурсів системи. Задача постачальника-споживача є однією з рішень для даної проблеми, що дозволяє запобігти затратам пам'яті, втраті інформації, а також ефективно забезпечує спосіб передачі даних.

Опис алгоритму ефективної передачі даних

Даний алгоритм, до задачі обмеженого буфера, описує два потоки роботи: перший – це потік постачальника (в подальшому буде використано слово "постачальник"), другий – потік споживач (в

подальшому буде використано слово “споживач”), які використовують спільний буфер, наперед встановленого розміру. Задачею постачальника є утворення фрагментів даних та передача їх до буферу – це циклічний процес, який буде виконуватись певну кількість раз, одночасно з цим споживач забирає дані з буфера, цим самим його і спорожнює, для подальшого використання даних. Отже, завданням є не допустити запису постачальником в повний буфер і не дозволити споживачу забирати дані з пустого буфера.

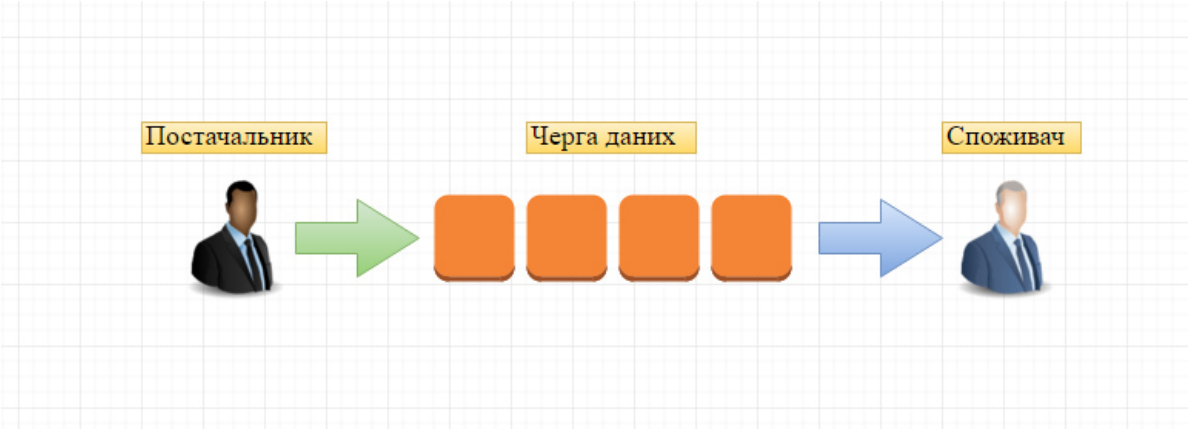

Рисунок 1 - Шаблон постачальник-споживач

На рис. 1 зображено абстракцію даної задачі, де постачальник забезпечує даними проміжний масив, звідки споживач буде їх забирати. В даному прикладі використано шаблон з одним постачальником та одним споживачем, хоча їх може бути більше.

Роботу даного шаблону забезпечують свого роду затримки. Дані затримки будуть ставити споживача в режим очікування, поки постачальник записує дані в буфер, і навпаки – постачальник буде чекати, поки не спорожниться буфер. Використання затримок зумовлює застосування так званих семафорів, які блокують потік до наступного використання. Вище згадані семафори є нічим іншим, як “замком” для даного потоку. Розрізняють такі види замків: mutex, recursive lock, read-write lock, distributed lock, spin lock double-checked lock, condition lock. Можна використовувати окремі з даних замків або можна їх комбінувати. Але потрібно бути обережним аби не спричинити deadlock (обидва потоки в режимі wait, очікують на роботу іншого) та livelock (невірне використання в одному потоці двох або більше замків). Слід правильно розташовувати замки, щоб не витрачати на їх блокування та розблокування багато ресурсів, саме це є основним для даної задачі.

Розглянемо повний алгоритм роботи передачі даних з одним постачальником та одним споживачем (два окремі класи, потоки), замість буфера використовується масив, як абстрактне вмістилище даних.

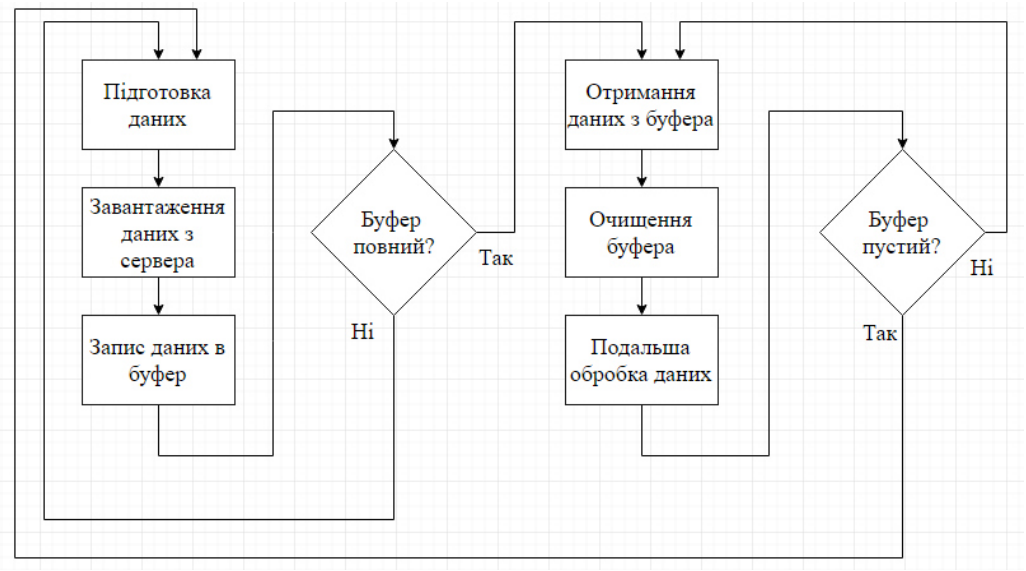

Рисунок 2 - Загальна схема алгоритму

На рис. 2 зображено опис роботи постачальника, що підготовляє дані для завантаження, завантажує їх із зовнішнього ресурсу та записує дані в буфер, після чого щоразу йде перевірка на те чи буфер повний, а споживач в свою чергу отримує дані з буфера, після чого його очищає, цей процес теж відбувається циклічно завдяки перевірці на те чи буфер пустий.

Отже, даний приклад показує ефективну роботу системи при передачі даних. Як вже було згадано, можливе використання декількох постачальників та споживачів, які будуть паралельно працювати, що забезпечить швидкодію.

Висновок

Використання потоків: постачальника та споживача дозволяє ефективно надсилати великі об'єми даних. Використання даного алгоритму надає такі переваги:

- економія ресурсів системи
- швидкодія передачі
- цілісність даних

Список використаних джерел

1. https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/Multithreading/ThreadSafety/ThreadSafety.html
2. https://developer.apple.com/library/mac/documentation/General/Conceptual/ConcurrencyProgrammingGuide/OperationQueues/OperationQueues.html
3. Design Patterns. Elements of Reusable Object-Oriented Software — СПб.: Питер, 2016. — 368 с.