Керування процесами, потоками (нитками)

Лаб 1
а) Тема: Процеси, потоки (нитки).
Завдання: Скласти програму (процес), який паралельно запускає два фонові потоки (нитки). Використати синхронізацію потоків на спільному ресурсі.
Опис роботи програми:
При запуску програми кнопкою "Пуск", одночасно паралельно запускаються два потоку Tthread1 і TThread2, які намагаються встановити "бігунок" в свою позицію (1-в позицію 10, 2-й в позицію 90)
У потоків можна змінювати пріоритети, і в залежності від пріоритету перевага віддається або одному, або іншому потоку
При завершенні програми потоки знищуються
б) Тема: Управління процесами, потоками (нитками) в критичній секції за допомогою блокуючої змінної (найпростішого семафора)
У завданні використовувати програму завдання 1
У програмі зробити наступні зміни:
Ввести глобальну змінну для семафора типу integer
Розмістити кнопки ПУСК 1 і ПУСК 2 для запуску першого і другого потоків (ниток), перед цим встановлюється семафор в положення "зайнято"
Розмістити кнопки СТОП 1 і СТОП 2 для зупинки першого і другого потоків (ниток), встановлюється семафор в положення "вільно"
Кнопка ПУСК 1 задає для першого потоку найнижчий пріоритет
Кнопка ПУСК 2 задає для другого потоку найвищий пріоритет
Опис роботи програми
Потоки запускаються послідовно. Якщо працює один з потоків, то другий неможливо запустити, тому що критична секція зайнята та відображає повідомлення "Зайнято потоком"
Кнопка Стоп звільняє критичну секцію і знищує поточний потік.
Кнопка ПУСК запускає потік і блокує натискання кнопки СТОП іншого потоку
Правильна робота програми полягає в наступному: Кнопка ПУСК 1 встановлює бігунок в положення 10, там він і залишається поки не натискаємо кнопку ПУСК 2, яка встановлює його в стан 90
Звернути увагу! Семафор (блокуюча змінна) - глобальна змінна, доступна обом потокам, отже вони працюють в одному адресному просторі (даного процессу).
Якби семафор регулював взаємодія не потоків, а процесів, то він повинен бути глобальним по відношенню до них і таким чином перебувати в адресному просторі операційної системи, яка і управляє процесами

Лаб 2(використати один варіант з використанням самостійної реалізації синхронізатора, з стандартної бібліотеки та на мові Golang)
Тема: многопоточність.
а) **Перше завдання про Вінні-Пуха.** Бджоли, підрахувавши в кінці місяця збитки від наявності в лісі Вінні-Пуха, вирішили розшукати його і покарати. Для пошуків ведмедя вони поділили ліс на ділянки, кожен з яких прочісує одна зграя бджіл. У разі знаходження ведмедя на своїй дільниці зграя проводить показове покарання і повертається у вулик.
Якщо ділянку перевірено, а Вінні-Пух на ньому не виявлено, зграя також повертається у вулик. Потрібно створити багатопоточний додаток, що моделює дії бджіл. При вирішенні використовувати парадигму портфеля завдань.
б) **Перша військова завдання.** Темної-темної ночі прапорщики Іванов, Петров і Нечипорчук займаються розкраданням військового майна зі складу рідної військової частини. Будучи розумними людьми і відмінниками бойової та стройової підготовки, прапорщики ввели поділ праці: Іванов виносить майно зі складу, Петров вантажить його в вантажівку, а Нечипорчук

підраховує вартість майна. Потрібно скласти багатопоточний додаток, що моделює діяльність прапорщиків. При вирішенні використати парадигму «виробник-споживач» з активним очікуванням.

с) **Завдання про Шлях Кулака.** На сивих схилах Гімалаїв стоять два древніх буддистських монастиря: Гуань-Інь і Гуань-Янь. Щороку в день зішестя на землю боддісатви Араватті ченці обох монастирів збираються на спільне свято і показують своє вдосконалення на Шляху Кулака. Всіх тих, що змагаються ченців розбивають на пари, переможці пар б'ються потім між собою і так далі, до фінального поєдинку. Монастир, ченці якого перемогли у фінальному бою, забирає собі на зберігання статую боддісатви. Реалізувати багатопоточний додаток, що визначає переможця. В якості вхідних даних використовується масив, в якому зберігається кількість енергії Ци кожного ченця. При вирішенні використовувати принцип дихотомії.

Лаб 3(використати один варіант з використанням самостійної реалізації синхронізатора, з стандартної бібліотеки та на мові Golang)
Тема: Двійкові семафори: захист спільного доступу до пам'яті

а) **Друге завдання про Вінні-Пуха.** В одному лісі живуть n бджіл і один ведмідь, які використовують один горщик меду, місткістю N ковтків. Спочатку горщик порожній. Поки горщик не наповниться, ведмідь спить. Як тільки горщик заповнюється, ведмідь прокидається і з'їдає весь мед, після чого знову засинає. Кожна бджола багаторазово збирає по одній порції меду і кладе його в горщик. Бджола, яка приносить останню порцію меду, будить ведмедя. Створить багатопоточний додаток, моделюючий поведінку бджіл і ведмедя.

б) **Завдання про перукаря.** У тихому містечку є перукарня. Салон перукарні малий, ходити там може тільки перукар і один відвідувач. Перукар все життя обслуговує відвідувача. Коли в салоні нікого немає, він спить в кріслі. Коли відвідувач приходить і бачить сплячого перукаря, він буде його, сідає в крісло і спить, поки перукар зайнятий стрижкою. Якщо відвідувач приходить, а перукар зайнятий, то він встає в чергу і засинає. Після стрижки перукар сам проводжає відвідувача. Якщо є інші відвідувачі які очікують, то перукар будить одного з них і чекає поки той сяде в крісло перукаря і починає стрижку. Якщо нікого немає, він знову сідає в своє крісло і засинає. Створити багатопоточний додаток, що моделює робочий день перукарні.

с) **Завдання про курців.** Є три процесу-курця і один процес-посередник. Курець безперервно скручує сигарети і курить їх. Щоб скрутити цигарку, потрібні тютюн, папір і сірники. У одного процесу-курця є тютюн, у другого - папір, а у третього - сірники. Посередник кладе на стіл по два різних випадкових компонента. Той процес-курець, у якого є третій компонент, забирає компоненти зі столу, скручує цигарку і курить. Посередник до чекає, поки курець не скінчить курити, потім процес повторюється. Створити багатопоточний додаток, що моделює поведінку курців і посередника. При вирішенні завдання використати семафори.

Лаб4(використати один варіант з використанням самостійної реалізації синхронізатора, з стандартної бібліотеки та на мові Golang)
Тема: Блокування читання-запису
а) **Завдання про читачів і письменників.** Базу даних поділяють два типи процесів - читачі та письменники. Читачі виконують транзакції, які переглядають записи бази даних, транзакції письменників і переглядають і змінюють записи.
Створити багатопоточний додаток, що працює із загальним файлом.
Для захисту операцій із загальним файлом використовувати блокування читання-запису. Файл містить послідовність записів виду: Ф.І.О.1 - телефон1, Ф.І.О.2 - телефон2 ... Повинні працювати наступні потоки:
1) потоки, що знаходять телефони за вказаним прізвищем;

2) потоки, на які ходять П.І.Б. за вказаним телефоном;
3) потоки, що видаляють і додають записи в файл.

б) **Питання про сад.** Створити багатопоточний додаток, що працює із загальним двомірними масивом. Для захисту операцій з загальним масивом використовувати блокування читання-запису. Двовимірний масив описує сад. У додатку повинні працювати такі потоки:
1) потік-садівник стежить за садом і поливає зів'ялі рослини;
2) потік-природа може довільно змінювати стан рослин;
3) потік-монітор1 періодично виводить стан саду в файл (не стираючи попередній стан);
4) потік-монітор2 виводить стан саду на екран.

с) **Питання про автобус**. Створити багатопоточний додаток, що працює із загальним графом. Для захисту операцій з графом використовувати блокування читання-запису.
Граф описує множину міст і множину рейсів автобусів від міста А до міста Б із зазначенням ціни квитка (за замовчуванням, якщо рейс від А до Б, він йде і від Б до А, з однаковою ціною). У додатку повинні працювати наступні потоки:
1) потік, що змінює ціну квитка;
2) потік, що видаляє і додає рейси між містами;
3) потік, що видаляє старі міста і додає нові;
4) потоки, що визначають чи є шлях від довільного міста А до довільного міста Б, і яка ціна такої поїздки (якщо прямого шляху немає, то знайти будь-який шлях з існуючих)

Лаб 5(використати один варіант з використанням самостійної реалізації синхронізатора, з стандартної бібліотеки та на мові Golang)
Тема: бар'єри
а) **Завдання про новобранців.** Множині новобранців дається команда «наліво» або «направо». Всі новобранці намагаються виконати наказ, але проблема в тому, що вони не знають де право, а де ліво. Отже, кожен новобранець повертається або направо, або наліво. Якщо новобранець повернувся і бачить, що його сусід стоїть до нього спиною, він вважає, що все зробив вірно. Якщо ж вони стикаються віч-на-віч, то обидва вважають, що помилилися, і розгортаються на 180 градусів. Створити багатопоточний додаток, що моделює поведінку множини новобранців, поки він не прийде до стаціонарного стану. Кількість новобранців ≥ 100. Окремий потік відповідає за частину ладу не менше 50 новобранців.

б) Створити додаток з чотирма потоками. Кожен потік працює з власної рядком. Рядки можуть містити тільки символи A, B, C, D.
Потік може поміняти символ A на C або C на A або B на D або D на B. Потоки зупиняються коли загальна кількість символів A і B стає рівним хоча б для трьох рядків.

с) Створити додаток з трьома потоками. Кожен потік працює зі своїм масивом, потоки перевіряють суму елементів свого масиву з сумами елементів інших потоків і зупиняються, коли всі три суми рівні між собою. Якщо суми не рівні , кожен потік додає одиницю до одного елементу масиву або віднімає одиницю від одного елемента масиву, потім знову перевіряє умова рівності сум. На момент зупинки всіх трьох потоків, суми елементів масивів повинні бути однакові.

Лаб 6(графічна реалізація з використанням подвійного буфера окремі потоки для обчислення та малювання)

Клітинний автомат: гра "Життя"

Основна ідея - розділити простір на окремі клітини. Кожна клітина - це кінцевий автомат. Після ініціалізації всі клітини спочатку роблять один перехід в новий стан, потім другий перехід і т.д. Результат кожного переходу залежить від поточного стану клітини і її сусідів.

Дано двомірне поле клітин. Кожна клітина або містить організм (жива), або порожня (мертва). У цьому завданні кожна клітина має вісім сусідів, які розташовані зверху, знизу, зліва, справа і по чотирьом діагоналям від неї. У клітин в кутах по три сусіда, а на кордонах - по п'ять.

Гра "Життя" відбувається наступним чином. Спочатку поле ініціалізується. Потім кожна клітина перевіряє стан своє і своїх сусідів і змінює свій стан у відповідності з наступними правилами.

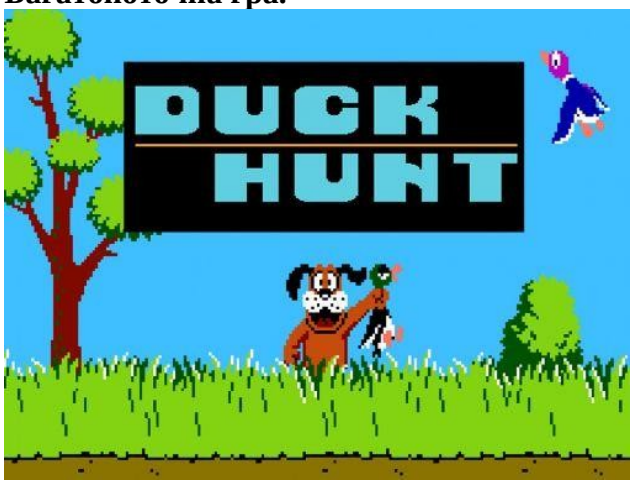
- *Жива клітина, біля якої менше двох живих клітин, вмирає від самотності.*
- *Жива клітина, біля якої є дві або три живі клітини, виживає ще на одне покоління.*
- *Жива клітина, біля якої знаходиться більше трьох живих клітин, вмирає від перенаселення.*
- *Мертва клітка, поруч з якою є рівно три живих сусіда, оживає.*

Цей процес повторюється певна кількість кроків (поколінь).
Див. Також http: //life.written.ru/game_of_life_review_by_gardner

а) Передбачити розвиток цивілізації (живі клітини) в декількох потоках, пояснити обраний варіант ефективного розподілу клітин за потокам

б) Одночасно стартувати кілька цивілізацій різними кольорами при конкуренції за ресурси (клітини) використовувати атомарні операції, семафор и або монітори, при зміні стану клітини.

Лаб7
Багатопоточна гра.



Орієнтовний варіант гри: https://www.youtube.com/watch?v=nnEhEKR_Rs4

а) Качки (кілька з різною швидкістю і напрямком) в окремий и n потоках. Збивання відбувається після натискання кнопки миші по качці.

б) Додатково потік з мисливцем який рухається в нижній частині при пострілі (space) вилітає куля. Збивання відбувається при перетині кулі з об'єктом Качки.

Лаб8

Обчислення на кластері з використанням MPI .
Створити додаток в якому обчислюється паралельне множення матриць .

Последовний алгоритм
Алгорітм1 -стрічкова схема
Алгорітм2 -метод Фокса
Алгорітм3 -метод Кеннона

Розрахунки по кожному алгоритму з робити у вигляді таблиці послідовний (1 потік), 2 потоку, 4 потоки.

розмірність	**послідовний (1потік),час**	**2 потоки**			**4 потоки**		
		час	Sp	Ep	час	Sp	Ep
100							
1000							
5000							

Лаб9
Написати програму з Лаб 8 (Алгоритм 1 - стрічкова схема) використовуючи функції що дозволяють автоматично розпаралелити код (заміряти час виконання) :

1. java.util.concurrent. ForkJoinPool
   http://www.h-online.com/developer/features/The-fork-join-framework-in-Java-7-1762357.html
   http://docs.oracle.com/javase/7/docs/api/java/util/concurrent/package-tree.html
   http://www.oracle.com/technetwork/articles/java/fork-join-422606.html
2. Multi-Core Parallel Programming in **Go (GO)**
   https://golang.org/doc/effective_go.html#concurrency
   https://www.youtube.com/watch?v=f6kdp27TYZs
   https://www.youtube.com/watch?v=QDDwwePbDtw
3. **Intel Threading Building Blocks** для розробки багатопоточного ПО
   https://www.threadingbuildingblocks.org/docs/help/tbb_userguide/
4. OpenMP паралельний регіон #pragma omp parallel for
   https://software.intel.com/ru-ru/articles/getting-started-with-openmp
5. TPL (C#) паралельний цикл Parallel.for;
   https://msdn.microsoft.com/ru-ru/library/dd460717%28v=vs.110%29.aspx