Микропроект № 1 по дисциплине
"Архитектура вычислительных систем" (осень-зима 2020).
Требования к оформлению.

Необходимо разработать и отладить многопоточную программу с использованием семафоров и (или) условных переменных в соответвии с выданным заданием. Результаты работы оформляются в виде пояснительной записки. Оформление должно включать титульный лист в котором должны отразиться: ВУЗ, департамент, название разработки, данные о студенте (ФИО, номер группы/подгруппы). Меня интересует не столько формальная сторона оформления титульного листа, сколько корректность и полнота представленной информации.

Последний срок сдачи работы **13.12.2020 г.** (воскресенье 23:59:59). Сдача после этого срока уменьшает оценку на 1 балл с каждым днем задержки.

В пояснительной записке (ПЗ) необходимо привести текст задания, описать подробно сценарий взаимодействия субъектов (объектов программы) на уровне тех понятий, которые используются при описании задачи, а также подробный протокол их взаимодействия. Эта информация впоследствии должна быть перенесена на потоки и выводимые в консоль сообщения об их текущем состоянии.

Список используемых источников должен содержать информацию, используемую при выполнении работы, а не просто формальную отписку. Если таких источников информации нет, то следует написат, что работа реализована впервые в мировой практике.

Исходные тексты программы в ПЗ включать не нужно. Они должны быть выложены отдельно. Учитывая небольшой размер кода достаточно сформировать весь исходный текст в одном файле. При реализации проекта предлагается использовать библиотеку **Posix thread (pthread)** или **стандартную библиотку потоков (thread)** языка программирования C++. Лучше **не использовать** Windows API, так как в этом случае может сильно замедлиться процесс проверки задания из-за отсутствия под рукой соответствующей операционной системы.

Разработанная программа должна запускаться из командной строки. Необходимые исходные данные могут быть аргументами командной строки или задаваться в диалоге после запуска. При наличии аргументов командной строки необходимо описать формат команды в ПЗ, а в программе, в случе некорректного ввода этих аргументов, предусмотреть вывод сообщения, описывающего требуемый формат.

Если программа использует большие по объему входные данные, то их предлагается вводить из заранее подготовленных файлов или порождать с использованием генератора случайных чисел (я не вижу больших данных в

условиях задач, но вдруг...). Генератор случайных чисел рекомендуется использовать при задании необходимых временных интервалах, предоставляя в протоколе вывода (при целесообразности) информацию о соответствующих задержках (возможно масштабируя их к масштабу модельного времени). При этом диапазон порождаемых данных должен быть в разумных для восприятия пределах и учитывать специфику задачи. Не нужно охватывать весь диапазон, определяемый разрядностью машинного слова.

Обратите внимание на юзабилити. При работе программа должна выводит информацию в виде протокола происходящих событий, однозначно определяя номер выполняемого потока и происходящее событие. При этом следует таким образом сформулировать и идентифицировать выводимые программой действи, чтобы они были в терминах предметной области. То есть, не выводить, например, «поток 1 обратился к элементу массива A[5]». Вместо этого, в соответствии со сценарием, определяемым задачей сформулировать работу потока в терминах предметной области. Например: «Время 20: Пират 1 приступил к обследованию квадрата № 5». В многопоточном программировании для лучшего понимания протокола работы предпочтительнее выводит информацию о событиях, а не самом процессе. То есть точках, когда процесс начался и когда он завершен («Время 25: Пират 1 не нашел клад в квадрате № 5» - это событие, определяющее конец процесс поиска). Обратите внимание, что задачи посвящены взаимодействию процессов. Поэтому наглядное отображение протокола является важным моментом. Понятно, что время масштабировать не нужно, так как оно отражает реальные моменты запуска событий.

Помните, что параллельные программы могут вести себя непредсказуемо. Поэтому внимательно отнеситесь к синхронизации потоков. Но не переусердствуйте. Не стоит сводить программу к чисто последовательным вычислениям без особой необходимости. Что касается периода работы программы. Практически во всех задачах отсутствует понятие ее завершения. То есть выполнение может продолжаться достаточно долго. Следует предусмотреть ограничение, например по числу итераций или по времени выполнения программы. Для этого можно использовать системные функции управления временем. Так как протокол имитирует взаимодействие, целесообразно вставить временные задержки, позволяющие читать на экране последовательность действий. Как вариант можно отключать (обнулять) эти задержки, если выставляются пределы на количество повторенний в качестве исходных данных. Тогда протокол можно перенаправлять в файл и изучать после завершения программы. Перенаправление можно специально не задавать, а использовать для этого перенаправление потока вывода а командной строке.

Пояснительную записку и программу необходимо выложить на **git**. Не стоит выкладывать на git упакованные архивы. Данная систем предназначена как раз для того, чтобы работать с исходными текстами. Она позволяет вносить изменения в ходе работы. При скачивании она сама формирует архивы или позволяет полностью синхронизировать проекты. Информацию о готовности программы и ссылку на репозиторий выслать преподавателю после того как убедитесь что все работает и обеспечивает формирование информации о работе программы в удобном виде.

Варианты задания определяются в соответствии с ранее высланными списками групп. При числе заданий меньшем числа студентов в группе номер определяется путем остатка от деления текущего номера в группе на число вариантов. Для одинаковых заданий (по всем группам) будет дополнительно проводиться сравнительнная оценка оригинальности протокола взаимодействия потоков и качество программы. Но при этом желательно не переусердствовать. Помните о принципе KISS...

Варианты заданий

1. *Задача о парикмахере.* В тихом городке есть парикмахерская. Салон парикмахерской мал, ходить там может только парикмахер и один посетитель. Парикмахер всю жизнь обслуживает посетителей. Когда в салоне никого нет, он спит в кресле. Когда посетитель приходит и видит спящего парикмахера, он будет его, садится в кресло и спит, пока парикмахер занят стрижкой. Если посетитель приходит, а парикмахер занят, то он встает в очередь и засыпает. После стрижки парикмахер сам провожает посетителя. Если есть ожидающие посетители, то парикмахер будит одного из них и ждет пока тот сядет в кресло парикмахера и начинает стрижку. Если никого нет, он снова садится в свое кресло и засыпает до прихода посетителя. Создать многопоточное приложение, моделирующее рабочий день парикмахерской.

2. *Задача о Винни-Пухе или правильные пчелы.* В одном лесу живут n пчел и один медведь, которые используют один горшок меда, вместимостью H глотков. Сначала горшок пустой. Пока горшок не наполнится, медведь спит. Как только горшок заполняется, медведь просыпается и съедает весь мед, после чего снова засыпает. Каждая пчела многократно собирает по одному глотку меда и кладет его в горшок. Пчела, которая приносит последнюю порцию меда, будит медведя. Создать многопоточное приложение, моделирующее поведение пчел и медведя.

3. *Задача о читателях и писателях.* Базу данных разделяют два типа процессов – читатели и писатели. Читатели выполняют транзакции, которые просматривают записи базы данных, транзакции писателей и просматривают и изменяют записи. Предполагается, что в начале БД находится в

непротиворечивом состоянии (т.е. отношения между данными имеют смысл). Каждая отдельная транзакция переводит БД из одного непротиворечивого состояния в другое. Для предотвращения взаимного влияния транзакций процесс-писатель должен иметь исключительный доступ к БД. Если к БД не обращается ни один из процессов-писателей, то выполнять транзакции могут одновременно сколько угодно читателей. Создать многопоточное приложение с потоками-писателями и потоками-читателями. Реализовать решение, используя семафоры.

4. *Задача об обедающих философах.* Пять философов сидят возле круглого стола. Они проводят жизнь, чередуя приемы пищи и размышления. В центре стола находится большое блюдо спагетти. Спагетти длинные и запутанные, философам тяжело управляться с ними, поэтому каждый из них, что бы съесть порцию, должен пользоваться двумя вилками. К несчастью, философам дали только пять вилок. Между каждой парой философов лежит одна вилка, поэтому эти высококультурные и предельно вежливые люди договорились, что каждый будет пользоваться только теми вилками, которые лежат рядом с ним (слева и справа). Написать многопоточную программу, моделирующую поведение философов с помощью семафоров. Программа должна избегать фатальной ситуации, в которой все философы голодны, но ни один из них не может взять обе вилки (например, каждый из философов держит по одной вилки и не хочет отдавать ее). Решение должно быть симметричным, то есть все потоки-философы должны выполнять один и тот же код.

5. *Задача о каннибалах.* Племя из n дикарей ест вместе из большого горшка, который вмещает m кусков тушеного миссионера. Когда дикарь хочет обедать, он ест из горшка один кусок, если только горшок не пуст, иначе дикарь будит повара и ждет, пока тот не наполнит горшок. Повар, сварив обед, засыпает. Создать многопоточное приложение, моделирующее обед дикарей. При решении задачи пользоваться семафорами.

6. *Задача о курильщиках.* Есть три процесса-курильщика и один процесс-посредник. Курильщик непрерывно скручивает сигареты и курит их. Чтобы скрутить сигарету, нужны табак, бумага и спички. У одного процесса-курильщика есть табак, у второго – бумага, а у третьего – спички. Посредник кладет на стол по два разных случайных компонента. Тот процесс-курильщик, у которого есть третий компонент, забирает компоненты со стола, скручивает сигарету и курит. Посредник дожидается, пока курильщик закончит, затем процесс повторяется. Создать многопоточное приложение, моделирующее поведение курильщиков и посредника. При решении задачи использовать семафоры.

7. *Военная задача.* Анчуария и Тарантерия – два крохотных латиноамериканских государства, затерянных в южных Андах. Диктатор Анчуарии, дон Федерико, объявил войну диктатору Тарантерии, дону Эрнандо. У обоих диктаторов очень мало солдат, но очень много снарядов для минометов, привезенных с последней американской гуманитарной помощью. Поэтому армии обеих сторон просто обстреливают наугад территорию противника, надеясь поразить что-нибудь ценное. Стрельба ведется по очереди до тех пор, пока либо не будут уничтожены все цели, либо стоимость потраченных снарядов не превысит суммарную стоимость всего того, что ими можно уничтожить. Создать многопоточное приложение, моделирующее военные действия.

8. *Задача о читателях и писателях-2 («грязное чтение»).* Базу данных разделяют два типа потоков – читатели и писатели. Читатели выполняют транзакции, которые просматривают записи базы данных, транзакции писателей и просматривают и изменяют записи. Предполагается, что в начале БД находится в непротиворечивом состоянии (т.е. отношения между данными имеют смысл). Транзакции выполняются в режиме «грязного чтения», то есть процесс-писатель не может получить доступ к БД только в том случае, если ее занял другой процесс-писатель, а процессы-читатели ему не мешают. Создать многопоточное приложение с потоками-писателями и потоками-читателями. Реализовать решение, используя семафоры, и не используя блокировки чтения-записи.

9. *Задача о читателях и писателях-3 («подтвержденное чтение»).* Базу данных разделяют два типа процессов – читатели и писатели. Читатели выполняют транзакции, которые просматривают записи базы данных, транзакции писателей и просматривают и изменяют записи. Предполагается, что в начале БД находится в непротиворечивом состоянии (т.е. отношения между данными имеют смысл). Каждая отдельная транзакция переводит БД из одного непротиворечивого состояния в другое. Транзакции выполняются в режиме «подтвержденного чтения», то есть процесс-писатель не может получить доступ к БД в том случае, если ее занял другой процесс-писатель или процесс-читатель. К БД может обратиться одновременно сколько угодно процессов-читателей. Процесс читатель получает доступ к БД, даже если ее занял процесс-писатель. Создать многопоточное приложение с потоками-писателями и потоками-читателями. Реализовать решение, используя семафоры, и не используя блокировки чтения-записи.

10. *Задача о супермаркете.* В супермаркете работают два кассира, покупатели заходят в супермаркет, делают покупки и становятся в очередь к случайному кассиру. Пока очередь пуста, кассир спит, как только появляется покупатель, кассир просыпается. Покупатель спит в очереди, пока не

подойдет к кассиру. Создать многопоточное приложение, моделирующее рабочий день супермаркета.

11. *Задача о магазине.* В магазине работают три отдела, каждый отдел обслуживает один продавец. Покупатель, зайдя в магазин, делает покупки в произвольных отделах, и если в выбранном отделе продавец не свободен, покупатель становится в очередь и засыпает, пока продавец не освободится. Создать многопоточное приложение, моделирующее рабочий день магазина.

12. *Задача о больнице.* В больнице два врача принимают пациентов, выслушивают их жалобы и отправляют их или к стоматологу или к хирургу или к терапевту. Стоматолог, хирург и терапевт лечат пациента. Каждый врач может принять только одного пациента за раз. Пациенты стоят в очереди к врачам и никогда их не покидают. Создать многопоточное приложение, моделирующее рабочий день клиники.

13. *Задача о гостинице.* В гостинице 30 номеров, клиенты гостиницы снимают номер на одну ночь, если в гостинице нет свободных номеров, клиенты устраиваются на ночлег рядом с гостиницей и ждут, пока любой номер не освободится. Создать многопоточное приложение, моделирующее работу гостиницы.

14. *Задача о гостинице-2 (умные клиенты).* В гостинице 10 номеров с ценой 200 рублей, 10 номеров с ценой 400 рублей и 5 номеров с ценой 600 руб. Клиент, зашедший в гостиницу, обладает некоторой суммой и получает номер по своим финансовым возможностям, если тот свободен. Если среди доступных клиенту номеров нет свободных, клиент уходит искать ночлег в другое место. Создать многопоточное приложение, моделирующее работу гостиницы.

15. *Задача о гостинице - 3 (дамы и джентльмены).* В гостинице 10 номеров рассчитаны на одного человека и 15 номеров рассчитаны на двух человек. В гостиницу приходят клиенты дамы и клиенты джентльмены, и конечно они могут провести ночь в номере только с представителем своего пола. Если для клиента не находится подходящего номера, он уходит искать ночлег в другое место. Создать многопоточное приложение, моделирующее работу гостиницы.

16. *Задача о клумбе.* На клумбе растет 40 цветов, за ними непрерывно следят два садовника и поливают увядшие цветы, при этом оба садовника очень боятся полить один и тот же цветок. Создать многопоточное приложение, моделирующее состояния клумбы и действия садовников. Для изменения состояния цветов создать отдельный поток.

17. *Задача о нелюдимых садовниках.* Имеется пустой участок земли (двумерный массив) и план сада, который необходимо реализовать. Эту задачу выполняют два садовника, которые не хотят встречаться друг с

другом. Первый садовник начинает работу с верхнего левого угла сада и перемещается слева направо, сделав ряд, он спускается вниз. Второй садовник начинает работу с нижнего правого угла сада и перемещается снизу вверх, сделав ряд, он перемещается влево. Если садовник видит, что участок сада уже выполнен другим садовником, он идет дальше. Садовники должны работать параллельно. Создать многопоточное приложение, моделирующее работу садовников. При решении задачи использовать мутексы.

18. *Задача о картинной галерее.* Вахтер следит за тем, чтобы в картинной галерее было не более 50 посетителей. Для обозрения представлены 5 картин. Посетитель ходит от картины к картине, и если на картину любуются более чем десять посетителей, он стоит в стороне и ждет, пока число желающих увидеть картину не станет меньше. Посетитель может покинуть галерею. Создать многопоточное приложение, моделирующее работу картинной галереи.

19. *Задача о Винни-Пухе - 3 или неправильные пчелы - 2.* N пчел живет в улье, каждая пчела может собирать мед и сторожить улей (N>3). Ни одна пчела не покинет улей, если кроме нее в нем нет других пчел. Каждая пчела приносит за раз одну порцию меда. Всего в улей может войти тридцать порций меда. Вини-Пух спит пока меда в улье меньше половины, но как только его становится достаточно, он просыпается и пытается достать весь мед из улья. Если в улье находится менее чем три пчелы, Вини-Пух забирает мед, убегает, съедает мед и снова засыпает. Если в улье пчел больше, они кусают Вини-Пуха, он убегает, лечит укус, и снова бежит за медом. Создать многопоточное приложение, моделирующее поведение пчел и медведя.

20. *Задача о болтунах.* N болтунов имеют телефоны, ждут звонков и звонят друг другу, чтобы побеседовать. Если телефон занят, болтун будет звонить, пока ему кто-нибудь не ответит. Побеседовав, болтун не унимается и или ждет звонка или звонит на другой номер. Создать многопоточное приложение, моделирующее поведение болтунов. Для решения задачи использовать мутексы.

21. *Задача о магазине - 2 (забывчивые покупатели).* В магазине работают два отдела, каждый отдел обладает уникальным ассортиментом. В каждом отделе работает один продавец. В магазин ходят исключительно забывчивые покупатели, поэтому каждый покупатель носит с собой список товаров, которые желает купить. Покупатель приобретает товары точно в том порядке, в каком они записаны в его списке. Продавец может обслужить только одного покупателя за раз. Покупатель, вставший в очередь, засыпает пока не дойдет до продавца. Продавец засыпает, если в его отделе нет покупателей, и просыпается, если появится хотя бы один. Создать многопоточное приложение, моделирующее работу магазина.

22. *Задача о программистах.* В отделе работают три программиста. Каждый программист пишет свою программу и отдает ее на проверку другому программисту. Программист проверяет чужую программу, когда его собственная уже написана. По завершении проверки, программист дает ответ: программа написана правильно или написана неправильно. Программист спит, если не пишет свою программу и не проверяет чужую программу. Программист просыпается, когда получает заключение от другого программиста. Если программа признана правильной, программист пишет другую программу, если программа признана неправильной, программист исправляет ее и отправляет на проверку тому же программисту, который ее проверял. Создать многопоточное приложение, моделирующее работу программистов.