РИС-ПЗ-1-2 Разработка многопоточного приложения

# Общие положения

Тема: Разработка многопоточного приложения

Общее описание цели выполнения: Основная цель - Приобрести опыт в разработке многопоточных приложений. Дополнительная цель – Получение опыта в синхронизации данных и/или операций между параллельно выполняющимися алгоритмами, который может быть распространен на РИС, выполняющуюся на нескольких хостах.

Задачи: 1) Получить практический навык использования средств синхронизации 2) Применить его к конкретной задаче 3) Научиться создавать демонстрационные примеры, результаты работы которых в явном виде демонстрируют требуемые заданием особенности (соответствующий этап жизненного цикла РИС – приемо-сдаточные испытания и подготовка к ним)

Предметная область: Индивидуальная для каждого студента. Предполагается, студент выберет одно из заданий раздела 3 и выполнит его

Форма отчета: проект выбранного языка программирования, включающий исходный код задачи и исполнимый модуль

Возможные платформы:

* MS Windows (XP и выше, рекомендовано MS Windows 10) – по умолчанию
* Linux
* OS X по согласованию с преподавателем

Возможные платформы и языки программирования:

* C++ - по умолчанию
* С# (.NET)
* Java
* Другие языки по согласованию с преподавателем

# Выполнение задания

Студент выбирает задание из списка, приведенного в разделе 3.

Определяет целевую платформу и язык программирования.

Создает программу, реализующую поведение, описанное в задании.

Предоставляет преподавателю результат в виде проекта языка программирования, включая исходный код и исполнимый модуль.

# Задания

## Легкие задания

1. Основной поток создает дочерний. Родительский и дочерний потоки должны распечатать по десять строк текста.
2. Основной поток создает дочерний. Родительский и дочерний потоки должны распечатать по десять строк текста. Вывод родительского потока должен осуществляться строго производился после завершения дочернего.
3. Создать два или больше однотипных потоков (их тело должно быть одной и той же функцией или методом).
4. Создать поток, которому данные для работы передаются как параметр.
5. Создать дочерний поток, выполняющийся теоретически бесконечно. Через две секунды после запуска остановить его из основного потока принудительно (без обработчика завершения).
6. Создать дочерний поток, выполняющийся теоретически бесконечно. Через две секунды после запуска остановить его из основного потока. Перед завершением дочерний поток выводит сообщение о команде на остановку на экран (то есть имеется обработчик завершения).

## «Обычная» сложность

1. Родительский поток должен послать дочерней сигнал SIGINT. Дочерний поток должен перехватить этот сигнал и распечатать пришедший сигнал и свой идентификатор.
2. Два потока, каждый из которых читает данные из своего текстового файла, и пишет в другой. Оба пишут в один файл. Продемонстрировать, что данные их двух исходных файлов перемешиваются в результирующем файле произвольным образом.
3. Два потока, упорядочивающих одномерный массив своим способом (например, пузырек и еще один). Показать, что в результате их работы часть данных из исходного файла может быть повреждена.
4. «Обедающие философы». Пять философов сидят за круглым столом и едят вилками спагетти. Между ними философами лежат вилки таким образом, что каждые двое философов, сидящих рядом, пользуются общей вилкой. Спагетти едят при помощи двух вилок. Философ некоторое время размышляет, потом пытается взять вилки и принимается за еду. Съев некоторое количество спагетти, философ освобождает вилки и снова начинает размышлять. Еще через некоторое время он снова принимается за еду, и т.д., пока спагетти не кончатся. Если одну из вилок взять не получается, философ ждет, пока она освободится. Философы всегда берут сначала левую вилку, а потом правую. При некоторых обстоятельствах это может приводить к мертвой блокировке.
5. Основной поток создает дочерний. Родительский и дочерний потоки должны распечатать по десять строк текста. При этом вывод должен быть синхронизирован таким образом, чтобы родительский и дочерний потоки выводили строки строго по очереди.
6. Основной поток программы считывает вводимые с экрана строки и помещает их в начало связанного списка. При вводе пустой строки программа должна печатает текущее состояние списка. При вводе строки «exit» выполнение программы завершается. Дочерний поток пробуждается каждые пять секунд и сортирует список по алфавиту.
7. Создать два или больше потока источника данных (целых чисел) и один поток обработчик, выполняющий с этими данными работу (простое арифметическое действие). Организовать передачу данных между потоками.

## Повышенная сложность

1. Два потока, каждый из которых читает данные из своего текстового файла, и пишет в другой. Оба пишут в один файл. Продемонстрировать один или несколько способов синхронизации, позволяющей избежать «мусора» в результирующем файле.
2. Два потока, упорядочивающих одномерный массив своим способом (например, пузырек и еще один). Показать два или больше способа, которыми можно избежать повреждения данных из исходного файла.
3. Создать распараллеливающийся алгоритм сортировки одномерного массива.
4. Изменить «Обедающих философов» так, чтобы мертвых блокировок не происходило.
5. Основной поток программы считывает вводимые с экрана строки и помещает их в начало связанного списка. При вводе пустой строки программа должна печатает текущее состояние списка. При вводе строки «exit» выполнение программы завершается. Дочерний поток пробуждается каждые пять секунд и сортирует список по алфавиту, выполняя при этом не более заданного числа операций по упорядочиванию (то есть, не обязательно завершая сортировку полностью).
6. Создать три потока источника разнотипных данных и один поток обработчик, выполняющий с этими данными работу (простое арифметическое действие). При этом для выполнения одной финальной операции нужно получить строго по одному элементу данных каждого типа. Исключить потерю данных, блокируя работу потоков-источников.
7. Создать три потока источника разнотипных данных и один поток обработчик, выполняющий с этими данными работу (простое арифметическое действие). При этом для выполнения одной финальной операции нужно получить строго по одному элементу данных каждого типа. Исключить потерю данных, используя буферизацию данных.