 netology-code / map-homeworks Public

<> Code

Issues

Pull requests

Actions

Projects

Security








Insights

map-homeworks / 02 / 



 **EkaterinaNikitina88** UPD\_TXT 

c5facbb · last year 

Name	Name	Last commit date
 ..		
 Atomic.cpp	Add files via upload	last year
 Timer.h	Add files via upload	last year
 mutex1.cpp	Add files via upload	last year
map-homeworks / 02 /		<a href="#">↑ Top</a>
 mutex3.cpp	Add files via upload	last year
 mutex_1_5.cpp	Add files via upload	last year
 readme.md	UPD_TXT	last year

readme.md 

# Домашнее задание к занятию «Конкуренция, состояние гонки»

## Цель задания

1. Создать программы, защищённые от блокировок.
2. Научиться использовать мьютексы и атомарные операции.
3. Научиться избегать взаимных блокировок.

## Задание 1

## Атомарная очередь клиентов

Нужно модифицировать [задание 1 к первому уроку](#) так, чтобы счётчик клиентов был атомарным.

Все операции со счётчиков должны быть атомарными.

Проверьте работу различными способами упорядочения доступа к памяти.

## Задание 2

### Прогресс-бар

Создайте консольное приложение для имитации многопоточного расчёта.

Количество потоков, длина расчёта должны быть заданы переменными.

В консоль во время работы программы должны построчно для каждого потока выводиться:

- номер потока по порядку;
- идентификатор потока;
- заполняющийся индикатор наподобие прогресс-бара, визуализирующий процесс «расчёта»;
- после завершения работы каждого потока в соответствующей строке суммарное время, затраченное на работу потока.



Строки прогресс-баров каждого потока должны выводиться одновременно. Время появления каждого нового символа в строке прогресс-бара подберите так, чтобы процесс заполнения строки был виден. Пример работы программы [по ссылке](#).

### Дополнение к заданию 2\*

Во время очередной итерации «расчёта» симулируйте со случайной вероятностью возникновение ошибки (exception), которая не должна приводить к прекращению работы потока или программы. При этом этот факт должен визуализироваться отдельным цветом на прогресс-баре.

## Задание 3

### Защищённый обмен данными

- Создайте класс Data, содержащий в качестве полей скалярные данные и мьютекс.
- Создайте функцию swap, которая принимает ссылки на два объекта класса Data и обменивает их местами.

- В функциях нужно сначала захватить мьютексы обоих объектов, а затем выполнить обмен данными.
  - Реализуйте три варианта этой функции: при помощи `lock`, `scoped_lock` и `unique_lock`.
- 

## Правила приёма домашней работы

Чтобы сдать домашнее задание, прикрепите в личном кабинете ссылку на ваш репозиторий.

## Критерии оценки домашней работы

1. В личном кабинете прикреплена ссылка на репозиторий с кодом для заданий 1, 2 и 3.
2. В ссылке содержится код, который при запуске выполняет описанный в задании алгоритм.