1. Часы на Qt:

К заготовке в репозитории <https://github.com/Milonov-Grigory/practice_clock.git> :

- дописать функционал программы: отображение всех стрелок

- сделать отображение циферблата: метки для часов и минут.

- подписи для позиций часов: 3, 6, 9, 12

- сделать отображение времени в тестовом формате по центру виджета: HH::MM::SS

Затем кинуть pull-request на этот репозиторий с готовой л.р.

1. Умные указатели:

Спроектируйте структуру "бинарное дерево" с использованием умных указателей так, чтобы узел дерева содержал указатели на дочерние узлы (shared\_ptr) и на родительский узел (weak\_ptr). Продемонстрируйте корректное освобождение памяти при удалении узла.

1. Многопоточность <thread>:

Задача о пяти обедающих философах:

*“Пять философов приглашены на обед. Их посадили за круглый стол, перед каждым из них стоит блюдо спагетти, справа от блюда лежит вилка. Как полагается истинным философам, гости не столько едят, как предаются размышлениям. Когда философ вспоминает о еде, то он берет свою вилку, но вскоре обнаруживает, что есть спагетти одной вилкой невозможно. Если вилка соседа свободна, то он берет вилку соседа и ест, пока опять не погружается в размышления, освобождая обе вилки. Если же вилка соседа занята, то философ ожидает ее освобождения, не выпуская свою вилку”*

*Необходимо наисать программу, которая будет эмитировать данную ситуацию: 5 философов с состояниями “придаться размышлению”, “проголодаться”, “есть”. (Обычно философы придаются размышлениям, через некоторое время становятся голодны и могут есть, лишь захватив две вилки: свою и соседа.)*

*Нужно показать процесс захвата вилок, а также взаимовозникающих блокировок философов. Это предлагается сделать с помощью вывода в консоль таблицы состояний философов и вилок.*

*Здесь же предлагается познакомиться с термином “ресурсное голодание” и проиллюстрировать его на философах: ввести показатель времени, которое каждый философ голодает, и выводить предупреждение (в ту же таблицу состояний), если философ голоден уже слишком долго.*

*Также необходимо написать вариант, работающий более оптимально, чем представленный на лекции.*

Возможные варианты:

1. Каждый философ пытается захватить палочки по очереди и если одна палочка захвачена, а вторая - нет, то он может вернуть обе палочки на стол и подождать некоторое случайное время. (назовем его вариант Ethernet)

2. В зависимости от времени ожидания философ может вести себя более агрессивно: например, при превышении времени ресурсного голодания он может пытаться агрессивно захватывать ресурсы пока не получит их все и не утолит голод. После этого счетчик голода сбрасывается и он опять ведет себя вежливо - то есть не блокирует один ресурс надолго.

3. Ваш вариант какой-нибудь.

1. Многопоточность

Существует большой объём случайных данных: вектор на 10^6 элементов. Необходимо посчитать не нулевые биты в этом векторе. Предполагается, что используемая функция будет не “наивной”. Реализация таких есть по ссылке: <https://habr.com/ru/post/276957/>

Необходимо:

* 1. Реализовать подсчёт не нулевых битов тривиальным образом, т.е. в однопоточной программе
  2. Сделать функцию (accumulate), принимающую на вход количество потоков, которое можно открыть для решения задачи, которая, соответственно, решит эту задачу с помощью многопоточности. Здесь необходимо вывести некоторую таблицу, демонстрирующую зависимость между количеством открытых потоков и временем выполнения программы. При необходимости можно построить график этой зависимости.

Замечания:

* 1. В “наивной” реализации явно показать время работы: вывести временные отметки начала и конца работы. В многопоточной реализации это предполагается показать в таблице.
  2. Функция accumulate предполагает, что вектор данных будет разбиваться на более мелкие куски с учётом количества потоков, которое можно открыть. Т.е.: сколько потоков можно открыть, на столько кусков разбивается вектор данных, и для каждого из этих кусочков в отдельных потоках запускается функция подсчёта бит. Затем все возвращаемые результаты будут складываться.
  3. В демо-варианте на лк(у нас его НЕТ) использовалась библиотека <future>

Доп. задание:

Провести эксперименты с кусочками вектора в функции accumulate (с их размером), чтобы минимизировать время работы программы (и максимально использовать кеш-память соответственно).

1. (Непонятно, будет ли проверяться) Многопоточность <future>

Необходимо дополнить примеры с лекции(лежат в папке future) сценариями выбрасывания исключений (set\_exception).