# Практическое задание по теме: «Параллельное программирование с блокировками»

## Задание 1. thread\_local. Мьютекс.

Задана простая иерархия классов:

enum COLOR{RED, GREEN, BLUE};

class Shape{

protected:  
 COLOR m\_color;

…

public:

static Shape\* ShapeMaker(void);  
};

class Rect:public Shape{  
 int x1,y1,x2,y2;  
 …

};

class Circle:public Shape{  
 int x,y,r;  
 …

};

Shape\* Shape::ShapeMaker()

{

//Сгенерировать случайным образом:  
 COLOR c = …;

switch(rand()&1)

{

case 0:

{

//Сгенерировать случайным образом

int x1 = …, y1 = …, x2 = …, y2 = …;

new Rect(c, x1, y1, x2, y2);

break;

}

case 1:

{

//Сгенерировать случайным образом

int x = …, y = …, r = …;

new Circle(c, x, y, r);

}

}

Запустите несколько потоков. Каждому потоку разрешите создать заданное количество фигур (прямоугольников и кружков – заранее неизвестно каких) с помощью статического метода ShapeMaker. Все фигуры создаются только на время работы потока. Перед завершением каждого потока выведите все созданные в потоке фигуры на консоль (чтобы было понятно, какая это фигура).

Подумайте: каким образом и где нужно хранить совокупность фигур. Кто и когда будет эту совокупность формировать? Кто и когда будет освобождать динамически захваченную память?

## Задание 2. Асинхронный запуск потоков. Получение отложенного результата. Параллельная реализация std::accumulate()

Задача: сформировать среднее значение для последовательностей (vector, deque) с элементами типа int

1. Создайте и заполните значениями контейнер с элементами типа int (количество элементов и значения можно формировать случайным образом)
2. Исходя из количества вычислительных ядер, запустите посредством std::async()разумное количество потоков, каждый из которых должен получать сумму заданного диапазона
3. Полученные частичные результаты по мере готовности требуется тоже просуммировать

## Задание 3. Обработка исключений посредством future

Реализуйте в отдельном дочернем потоке прием целых значений от пользователя.

Если пользователь вводит НЕ целое, должно быть сгенерировано исключение.

Обработку исключения требуется реализовать в родительском потоке.

## Задание 4. Потокобезопасная очередь на базе условных переменных

Требуется обеспечить безопасную работу приложения, в котором:

* несколько потоков генерируют заявки (каждый со своим периодом и значением) и складывают заявки в очередь и сигнализируют о готовности ( потоки «клиенты»)
* а один поток «сервер» по мере готовности достает заявки из очереди и обрабатывает (имитация обработки – просто вывод значения на консоль)
* для потокобезопасной очереди реализовать операции копирования и присваивания