Бригадные задания

1. Усовершенствовать класс `CircularBuffer` и его итератор, разработанные при решении бригадного задания ЛР № 4, аналогично п. 1 общего задания. Итератор должен обеспечивать перемещение по буферу в обе стороны; разыменование итератора до первого вызова `moveNext()`, `++` или `--` считается ошибкой.
2. Создать коллекцию элементов типа **`int`** (не менее 100 элементов), используя один из контейнеров STL по своему выбору. Рассчитать среднее арифметическое и среднеквадратическое отклонение по элементам коллекции, для чего использовать алгоритм `std::for_each()` и λ-функцию с захватом переменных-результатов.
3. Разработать приложение, тестирующее производительность контейнеров STL `vector`, `list` и `forward_list` одной и той же функцией. Необходимо оценить удельные временные затраты для операций добавления и удаления элементов, доступа к элементам и объяснить полученные результаты.

   Указание. Замеры времени можно производить функцией `clock()` из заголовочного файла `<ctime>` или средствами STL из пространства имен `std::chrono`. Достаточную точность удастся получить при размерах контейнера порядка тысяч элементов и с повторением операций порядка сотен — тысяч раз.
4. Разработать шаблонный класс стека `Stack`, принимающий в качестве одного из параметров шаблона тип-контейнер для хранения элементов стека. Тип-контейнер реализует шаблонный интерфейс `ICollection<Type>`, класс стека также должен реализовывать этот интерфейс.
5. Разработать класс итератора по бесконечной последовательности случайных чисел, совместимый с STL. Для этого изучить подразделы 24.3 и 24.4 («Iterator primitives») стандарта C++11 и снабдить класс-итератор необходимыми членами. Сгенерировать набор случайных чисел алгоритмом `std::generate()`. Генерация случайных чисел должна осуществляться средствами STL, а именно — любыми из содержащихся в заголовочном файле `<random>`.
6. Реализовать шаблонный алгоритм `shuffle(Iterator first, Iterator last)`, перемешивающий значения в диапазоне итераторов [`first`; `last`) в случайном порядке. Итераторы поддерживают произвольный доступ и запись в элементы (random access iterator и output iterator). Генерация случайных чисел

должна осуществляться средствами STL, а именно — любыми из содержащихся в заголовочном файле `<random>`.

7. Усовершенствовать класс красно-черного дерева, созданный в ЛР № 4, применив в реализации «умные» указатели STL. Типы «умных» указателей (`std::unique_ptr`, `std::shared_ptr` или `std::weak_ptr`) необходимо выбрать по своему усмотрению и объяснить решение. Объекты класса должны остаться копируемыми.
8. Усовершенствовать класс `HashTable`, созданный в ЛР № 4, сделав его шаблонным `HashTable<TKey, TValue, THashFunctor>`: тип ключей и данных задается типами-параметрами шаблона. Третьим параметром шаблона является тип функтора, которому передается аргумент-ключ для расчета хэша:

```
TKey key;
THashFunctor calculator;
size_t hash = calculator(key);
```

9. Разработать класс шаблонного умного указателя. Параметром шаблона является тип данных `T`, адрес которых будет хранить указатель. Поведение объекта должно быть таким, чтобы его можно было использовать как указатель `T*`, передав в конструкторе реальный указатель на уже выделенный блок памяти. Освобождение памяти должно выполняться автоматически в деструкторе объекта. Поведение при копировании и перемещении реализуйте так, как сочтете нужным, но не запрещайте эти операции.
10. Разработать шаблонный класс буфера (динамического массива фиксированного размера) для элементов произвольного типа. Аргументом шаблона является тип элементов, которые будут хранится в буфере. Класс буфера должен иметь методы:
    1) `add()` — добавить элемент в конец буфера;
    2) `addRange()` — добавить в конец буфера элементы диапазона, заданного двумя итераторами;
    3) `begin()` — получить итератор на начало буфера;
    4) `clear()` — очистить буфер;
    5) `shift()` — освободить место, занимаемое *n* первыми элементами в буфере (*n* — параметр метода), и сдвинуть все оставшиеся элементы в буфере к его началу.
11. Разработать шаблонный класс, производящий расчет основных статистик (минимальное значение, максимальное значение, оценки медианы, мат. ожидания, дисперсии) отдельными методами. Параметром шаблона является тип итераторов,

задающих диапазон. Конструктор принимает в качестве аргументов итераторы на начало и на конец диапазона данных для расчетов.

12. Разработать шаблонный класс очереди. Первым параметром шаблона может быть один из типов-контейнеров STL, и в нем должны храниться элементы очереди. Вторым параметром является тип элементов, которые будут храниться в очереди. Параметром конструктора класса является наибольший размер очереди. У класса должны быть методы:
    1) `enqueue()` — добавить элемент в конец очереди;
    2) `enqueueRange()` — добавить элементы в конец очереди элементы из некоторого диапазона, заданного двумя итераторами;
    3) `extract()` — извлечь элемент с начала очереди;
    4) `getLength()` — получить количество элементов в очереди.

    В ситуации, когда происходит добавление элемента в заполненную очередь или извлечение элемента из пустой очереди, должно быть сгенерировано исключение.
13. Разработать шаблонную функцию, которая выполняет численное интегрирование методом Симпсона для некоторого диапазона данных. Первым аргументом шаблона является тип итератора по диапазону; в качестве параметров функция принимает итераторы на начало и конец этого диапазона. Вторым параметром шаблона является тип шага дискретизации, значение которого передается параметром функции.