1 Базовые задачи на шаблоны

1.1 Сортировка

Любую сортировку (возьмите из предыдущего семестра или напишите её с асимптотикой $O(n \ln n)$) напишите в стиле стандартной библиотеки со следующей сигнатурой

```
template< typename IteratorCategory, typename Compare >
void sort( IteratorCategory first, IteratorCategory last, Compare comp );
```

- first, last – итератора на начало и конец диапазона
- comp – экземпляр функции компаратора

Подумайте над тем, какую категорию итератора нужно взять для удачной реализации выбранной сортировки (отразите это в названии типа итератора). Продемонстрируйте, что сортировка работает на разных типах контейнера, а также покажите, для каких контейнеров её не получиться применить (подсказка: категория итератора должна быть неудовлетворительной). Также продемонтсрируйте, что вместо comp можно передавать обычную функцию, лямбду, функтор (класс с перегруженными круглыми скобками).

Hint: в страндартной библиотеке есть функция **std::iter_swap**, используйте её

С помощью библиотеки chrono измерьте время сортировки максимального возможно количества контейнеров, которые получиться запихнуть в вашу функцию сортировки. Напишите код, который измеряет время и выводит его в консоль. Кто быстрее сортируется: vector или deque?

1.2 Адаптер над контейнером

Написать очередь или стэк на одном из стандартных контейнеров. Контейнер задается шаблонным параметром (по умнолчанию можно использовать std::deque). Написать отдельную полную специализацию, если в качестве контейнера передается std::string, то есть предполагается очередь изи стек из char. Реализуйте базовые методы, top, pop, push, расчет длины и проверку на пустоту.

2 Вариабельные шаблоны

2.1 Рекурсивное инстанцирование

Напишите функцию, которая применяет какую-либо унарную операцию (например, печатает в стандартный поток) к каждому элементу в std::tuple. Использовать цикл for не поулчиться – функция std::get шаблонная по номеру элемента кортежа, который надо достать. Так что придется делать рекурсивное инстанцирование шаблонов с «откусываением типов» в пачке шаблонных параметров для кортежа. Посмотрите код с семинаров, на котором мы такие штуки проворачивали. Напишите кусок кода, который показывает, что ваша реализация работает.

3 Программирование времени компиляции

3.1 Декартово произведение

Пусть на этапе компиляции известно произвольное число массивов разной (но известной на этапе компиляции) длины: arr1, arr2, arr3 и т.д. Пусть все эти массивы содержат в общем случае разные типы данных. Реализуйте функцию, которая вычисляет декартово произведение этих массивов на этапе компиляции.

Дальше есть развилка

- В качестве результирующего контейнера возьмите несколько вложенных std::array<std::array и т.д. >, в качестве типа внутреннего массива возьмите std::tuple<type1, type2, и т.д.>, где typei – это тип элемента iго массива. При этом очевидно надо вычислить все элементы в декартовом прозведении на этапе компиляции.

- Реализуйте обычную функцию, которая вычисляет не все элементы, а только один конкретный с заданными индексами. При этом нужно сделать так, чтобы при неправильном количестве индексов, или при выходе индексов за диапазон программа падала на этапе компиляции, а не с сегфолтом на этапе исполнения. Этот вариант будет стоить меньше баллов, чем предыдущий, так как он проще

3.2 Линейный рекуррент

Напишите программу, которая на этапе компиляции вычисляет произвольный элемент из линейного рекурретна произвольной длины с произвольными коэффициентами. Линейным рекуррентом длины n называется последовательность значений, определяемых рекуррентно по формуле:

$$x_{n+k} = a_{n-1} * x_{n+k-1} + a_{n-2} * x_{n+k-2} + ... + a_1 * x_{k+1} + a_0 * x_k,$$

$x_0, ... x_{n-1}$ – некоторые начальные данные

$a_0, ... a_{n-1}$ – некоторые произвольные коэффициенты

Простейшим примером линейного рекуррента является последовательность чисел Фибоначчи. Ваше решение должно по начальным данным и коэффициентам уметь строить любой элемент линейного рекуррента.

4 Статический интерфейс

Здесь будет все просто: напишите статический интерфейс на адаптер и сортировку из первой части «Базовые задачи на шаблоны» в любой удобном стиле (через std::void_t или через static_assert), но не через концепты, ими пользоваться здесь нельзя. Продемонтрируйте, что код на работающих контейнерах работает, а на неработающих не работает.

5 Система оценивания

Всего баллов $2 + 3 + 3 + 5(2) + 3 + 2 = 18$, из которых в зачет пойдет 15, остальное приедет в качестве доп баллов в конце семестра.