Вариант 1

1) Разработать на базе класса std::vector шаблонный класс «отсортированный вектор» (использовать стандартные алгоритмы, значения сортируются от минимального к максимальному):

2) Создать конструктор, принимающий список инициализации std::initializer_list<T> (вызовите конструктор базового класса) и сортирующий содержимое вектора;

Т.е. ваш вектор должен уметь инициализироваться значениями вот таким способом - vector<int> myvect = {1, 5, 646, 1313, 1};

3) Перегрузить insert следующим образом: вставка значения в первую позицию, в которую можно вставить значение без изменения порядка следования объектов;

Пояснение: изначально insert принимает 2 аргумента – позиция, перед которой надо вставить новый элемент и значение нового элемента. Ваш insert долен принимать только значение нового элемента, а позицию вы сами определите.

Например, у вас был вектор из 2 элементов 3 и 5. Хотят добавить еще элемент 2. Вы должны поставить его в начало вектора, чтобы получилось 2, 3, 5. Теперь хотят добавить 4. Вы должны положить новый элемент перед элементом за значением 5, чтобы получилось 2, 3, 4, 5.

4) Реализовать метод, который проверяет количество элементов в векторе, делящихся нацело на введенное пользователем число. Использовать лямбда-выражение.

5) Реализовать метод, который выводит на экран все перестановки без повторений из содержимого данного вектора

Пояснение: У вас в векторе лежат элементы 1, 2, 3. Вы должны вывести:

1 2 3

1 3 2

2 1 3

2 3 1

3 1 2

3 2 1

6) Перегрузить дружественную функцию - оператор вывода данных в поток (использовать* std::copy)

Вариант 2

1) Разработать класс для хранения делителей натурального числа. Конструктор принимает само число, и инициализирует все делители (изменить их далее нельзя).

2) С использованием стандартного алгоритма std::find_if для поиска внутри контейнера реализовать дружественную функцию, которая для двух чисел находит наибольший общий делитель. Сделать это двумя способами:

а) используя std::bind

б) используя лямбда-выражения.

3) С использованием стандартного алгоритма реализовать метод, который находит среднее арифметическое делителей числа.

Вариант 3

1) Разработать шаблонный класс «Квадратное уравнение». Конструктор принимает коэффициенты (целые, либо дробные), и находит корни уравнения (изменить их далее нельзя).

2) С использованием стандартного алгоритма реализовать метод, который проверяет, есть ли среди корней введенное пользователем число.

3) С использованием стандартного алгоритма реализовать метод, который проверяет количество корней, которые меньше заданного пользователем числа. Использовать лямбда-выражение.

4) С использованием стандартного алгоритма реализовать метод, который возвращает все уникальные (различные по значению) корни уравнения.

5) С использованием стандартных алгоритмов, реализовать функцию, которая:

а) генерирует (std::generate) 30 случайных уравнений (с разными коэффициентами и с разными типами данных), получает корни от всех уравнений и собирает их в массив;

б) находит сумму корней всех уравнений;

в) сортирует корни по убыванию и выводит их на экран.