Семинар #7: Move-семантика и умные указатели. Домашнее задание.

В задачах (`1 - 5` включительно) нельзя использовать циклы. Их нужно решить, используя алгоритмы STL.

Задача 1. Горка

На вход программе подаётся n чисел. Найдите первый максимум среди этих чисел. Отсортируйте часть массива, которая идёт до этого максимума по возрастанию. А часть массива, которая идёт после первого максимума отсортируйте по убыванию.

вход	выход
`10` `5 2 1 5 6 8 6 4 3 8`	`1 2 5 5 6 8 8 6 4 3`

Задача 2. Обращение вектора строк

Напишите функцию, которая принимает на вход вектор строк и обращает сам вектор, а также каждую его строку.

аргумент	выход
`{"cat", "dog", "mouse", "elephant"}`	`{"tnahpele", "esuom", "god", "tac"}`
`{"a", "bc"}`	`{"cb", "a"}`

Задача 3. Проверка на верхний регистр

Напишите функцию, которая будет принимать на вход строку и проверять находится ли эта строка в верхнем регистре.

аргумент	выход
`"Cats and Dogs!"`	`false`
`"CATS AND DOGS!"`	`true`
`"ABc123!#?"`	`false`
`"ABC123!#?"`	`true`

Задача 4. Идентификатор

Напишите функцию `bool isIdentifier(std::string_view s)`, которая будет принимать на вход строку и проверяет является ли эта строка допустимым идентификатором в языке C++. Случаи, когда приходящая на вход строка является одним из ключевых слов языка C++, можно не рассматривать.

аргумент	выход
`"a"`	`true`
`"isIdentifier"`	`true`
`"_name123"`	`true`
`"hello world"`	`false`
`"123name"`	`false`
`"my-name"`	`false`
`"int"`	`true`

Задача 5. Передвинуть пробелы

Напишите функцию, которая будет принимать на вход строку по ссылке и передвигать все её пробелы в конец.

аргумент	выход
`"cats and dogs"`	`"catsanddogs   "`
`"cats and dogs"`	`"catsanddogs      "`

Задача 6. Шаблонный максимум

Напишите шаблонную функцию `maxElement`, которая должна будет принимать 2 итератора и возвращать максимальный элемент на диапазоне, задаваемом этими итераторами (как лучше вернуть элемент? по ссылке или по значению?). Элементы контейнера должны сравнимы с помощью оператора меньше (`<`). Протестируйте эту функцию на различных контейнерах (`std::vector`, `std::list`, `std::set`).

Задача 7. Шаблонный обмен соседних

Напишите шаблонную функцию `swapNeighbours`, которая должна будет принимать 2 итератора и менять местами пары соседних элементов на диапазоне, задаваемом входящими итераторами. Если в диапазоне нечётное количество элементов, то последний элемент должен остаться на месте. То есть, если диапазон содержал элементы `{10, 20, 30, 40, 50}`, то после исполнения этой функции элементы диапазона должны иметь вид `{20, 10, 40, 30, 50}` Для обмена элементов используйте функцию `std::swap`. Протестируйте эту функцию на различных контейнерах (`std::vector`, `std::list`, `std::forward_list`).

Задача 8. Поиск соседей

Напишите шаблонную функцию `bestNeighbours`, которая должна будет принимать 2 итератора и функциональный объект, принимающий 2 элемента. Функция должна находить такую пару соседних элементов, что результат применения функционального объекта к этой паре будет наибольшим. Функция должна возвращать итератор на первый элемент этой пары. Например, если у нас есть такой вектор:

```
std::vector<int> v {50, 10, 10, 20, 90, 30, 40, 60, 80, 20};
```

То если мы применим к нему функцию `bestNeighbours` вот так:

```
auto it = bestNeighbours(v.begin(), v.end(), [](int a, int b){return a + b;});
```

то функция должна вернуть итератор на восьмой элемент (60), потому что пара элементов 60 и 80 имеют наибольшую сумму. А если мы применим эту функцию вот так:

```
auto it = bestNeighbours(v.begin(), v.end(), [](int a, int b){return std::abs(a - b);});
```

то функция должна вернуть итератор на четвёртый элемент (20), потому что пара элементов 20 и 90 имеют наибольший модуль разности. Протестируйте эту функцию на различных контейнерах (`std::vector`, `std::list`, `std::set`, `std::forward_list`).