Контрольные вопросы:

- ☐ (5 б.) Что такое шаблон и какие разновидности шаблонов существуют?
- ☐ (5 б.) Каким образом осуществляется двухэтапная трансляция шаблона?
- ☐ (5 б.) Что предпочитает компилятор при перегрузке шаблона функции?
- ☐ (5 б.) В чем заключается особенность инстанцирования шаблонов классов?
- ☐ (5 б.) Когда необходимы полная или частичная специализации шаблона?

Упражнения:

- ☐ (50 б.) Реализуйте шаблон функции произвольного алгоритма сортировки со сложностью $O(N \log N)$. В качестве контейнера для сортируемой последовательностью рассмотрите следующие варианты: встроенный фиксированный массив и встроенный динамический массив. Это вынужденная мера, т.к. более общее решение основано на итераторах и контейнерах STL, с которыми мы познакомимся только в следующем семестре. При реализации шаблона функции, принимающей в качестве аргумента встроенный фиксированный массив, используется особый синтаксис, позволяющий автоматически вывести размер массива, благодаря чему не потребуется дополнительный аргумент для размера. Следующий код демонстрирует данный способ:

```
template < typename T, std::size_t N >
void print(T (&array) [N])
{
  for (auto i = 0U; i < N; ++i)
    std::cout << array[i] << std::endl;

  for (auto value : array)
    std::cout << value << std::endl;

  std::for_each(array, array + N, [](auto value)
  {
    std::cout << value << std::endl;
  });
}

int a[] = { 1,2,3,4 };

print(a);
```

  При использовании динамических встроенных массивов синтаксис остается прежним, за исключением типа:

```
template < typename T >
void sort(T * array, std::size_t size)
{
    // ...
}
```

  Предоставьте пользователю возможность задавать критерий сортировки. Для этого потребуется добавить дополнительный параметр шаблона и соответствующий ему дополнительный аргумент функции для передачи бинарного компаратора. В качестве примера можете посмотреть сигнатуру и реализацию стандартного алгоритма `std::sort`. Компаратор вызывается в цикле алгоритма посредством оператора круглые скобки.

- ☐ (50 б.) Реализуйте шаблон класса контейнера, являющегося оберткой вокруг встроенного динамического массива. Обеспечьте данный класс необходимыми пользовательскими конструкторами, полным комплектом копирующих и перемещающих операций, а также деструктором. В качестве основы можете использовать собственные наработки, подготовленные в рамках одного из предыдущих заданий, либо соответствующие фрагменты учебного класса String, рассмотренного на одном из предыдущих семинаров. Реализуйте операторы для записи и чтения значений, хранящихся в ячейках встроенного массива. Также реализуйте функцию-член resize, выполняющую изменение размера встроенного массива до указанного пользователем значения. Не забудьте про вспомогательные функции, такие как size и swap. Для удобства и лучшей читабельности кода используйте псевдонимы. За дополнительные баллы можете реализовать другие компоненты, например, обработку ошибок с помощью исключений или специализацию данного шаблона класса.