# Задание по программированию: Шаблон Paginator

### Условие

В лекции мы разработали функцию Head, которая позволяет пройтись циклом for по началу контейнера. В этой задаче мы сделаем шаг вперёд и разработаем шаблон Paginator, который разбивает содержимое контейнера на несколько страниц. Классический пример, когда такое может пригодиться на практике, — это распределение списка мобильных приложений по экранам телефона. Допустим, у нас есть вектор всех приложений нашего телефона и на одном экране мы можем разместить 20 иконок приложений. Тогда распределить приложения по экранам мы можем вот таким кодом:

vector<vector<Application>> DistributeAmongScreens(const vector<Application>& apps) {

  vector<vector<Application>> result;

  for (const auto& page : Paginate(apps, 20)) {

    result.push\_back({page.begin(), page.end()});

  }

  // result[0] - все приложения, которые попадают на первый экран,

  // result[1] - все приложения, которые попадают на второй экран и т.д.

  return result;

}

Заметьте, наш код получился коротким и элегантным. Нам не пришлось писать какой-то отдельный код для обработки последнего экрана, который может содержать меньше 20 приложений.

Итак, разработайте шаблон класса Paginator со следующими свойствами:

* он имеет один шаблонный параметр — тип итератора
* конструктор класса Paginator<Iterator> принимает три параметра:

1. Iterator begin
2. Iterator end — пара итераторов begin и end задают полуинтервал [begin; end), который мы будем нарезать на страницы
3. size\_t page\_size — размер одной страницы

* по объектам класса Paginator<Iterator> можно проитерироваться с помощью цикла range-based for
* класс Paginator<Iterator> имеет метод size\_t size() const, который возвращает количество страниц, на которые был разбит переданный контейнер
* сами страницы должны так же поддерживать итерацию с помощью range-based for и иметь метод size\_t size() const, возвращающий количество объектов в этой странице
* подробные примеры использования смотрите в юнит-тестах в шаблоне решения

Кроме того разработайте шаблон функции Paginate, которая принимает ссылку на контейнер и размер страницы, и возвращает объект класса Paginator<It>:

template <typename C> ??? Paginate(C& c, size\_t page\_size)

### Файл с заготовкой решения задачи

[paginator.cpp](https://d3c33hcgiwev3.cloudfront.net/-qGdtHYQEei8FQ6TnIKJEA_fb073cf0761011e887545d40e262c3fc_paginator.cpp?Expires=1617062400&Signature=B71osclM2mhU2cfn5cCeiEVNPZVdzyCOf2Cpm4Y4VeINL4s9dk~jGtwuveznekUVyYdafb6hWX1d8hlWD88qJmClqqtKdk3vNwePrJRaBtmdYyOxa1MK864Z5b7Ai~eHjigc9DOVHoqsfQ6a4voKgHBDIfNAJWKbfvAGm47x~Bo_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A)

Возможно, в шаблоне у вас будет некорректно работать вызов ASSERT\_EQUAL в функции TestLooping. Разберитесь, почему это происходит, внесите правку в свою локальную версию файла test\_runner.h, чтобы подобная ошибка не возникала в других задачах.

Подсказки

Для этой задачи есть набор подсказок, которые должны помочь вам с решением. Если вам не удаётся решить задачу и вы чувствуете, что у вас кончились идеи, вы можете ими воспользоваться. Но сначала обязательно попробуйте решить задачу без подсказок.

Подсказка 1

Представление одной страницы Для представления одной страницы идеально подходит шаблон IteratorRange , который мы написали в видео лекции. К нему только надо добавить метод size

Подсказка 2

Тип итератора в шаблоне Paginate В шаблоне Paginate мы принимаем контейнер, и по нему должны вывести тип итератора, чтобы проинстанцировать шаблон Paginator . Мы такое уже проделывали в видеолекции. Здесь нам очень поможет auto в качестве типа возвращаемого значения

Подсказка 3

Как обеспечить итерирование по страницам? Нам нужно сделать так, чтобы у класса Paginator были методы begin/end , возвращающие итератор на страницу. Самый простой способ сделать это — сложить все страницы в vector<IteratorRange> .