**std::integer\_sequence**.

Определен в заголовке <utility>.

template<class T, T… Ints> class integer\_sequence;

Данный шаблонный класс std::integer\_sequence предоставляет на этапе компиляции последовательность целых чисел. Когда используется в качестве аргумента для шаблонной функции, пакет параметров шаблона Ints может быть выведен и использован в pack expansion.

T – целочисленный тип, используемый для представления элементов последовательности.

…Ints – пакет параметров шаблона, представляющий последовательность.

**Вспомогательные шаблоны.**

Вспомогательный псевдоним std::index\_sequence определен для общего случая, когда T есть std::size\_t.

template<[std::size\_t](http://en.cppreference.com/w/cpp/types/size_t)... Ints>  
using index\_sequence = std::integer\_sequence<[std::size\_t](http://en.cppreference.com/w/cpp/types/size_t), Ints...>;

Вспомогательный псевдоним std::make\_integer\_sequence определен для упрощения создания типов std::integer\_sequence и std::index\_sequence для последовательсности целых 0, 1, 2, …, N-1.

template<class T, T N>   
using make\_integer\_sequence = std::integer\_sequence<T, */\*a sequence 0, 1, ..., N-1\*/* >;

template<[std::size\_t](http://en.cppreference.com/w/cpp/types/size_t) N>  
using make\_index\_sequence = make\_integer\_sequence<[std::size\_t](http://en.cppreference.com/w/cpp/types/size_t), N>;

**Как это работает.**

NB: пояснение принципа работы шаблона make\_index\_sequence. Не совершенно точная стандартная индексная последовательность, которая получается от std::integer\_sequence, но фиксируя тип std::size\_t вы можете получить полноценную пару indexSequence / makeIndexSequence с помощью следующего кода:

// index sequence only

template <std::size\_t ...>

struct indexSequence

{ };

template <std::size\_t N, std::size\_t ... Next>

struct indexSequenceHelper : public indexSequenceHelper<N-1U, N-1U, Next...>

{ };

template <std::size\_t ... Next>

struct indexSequenceHelper<0U, Next ... >

{ using type = indexSequence<Next ... >; };

template <std::size\_t N>

using makeIndexSequence = typename indexSequenceHelper<N>::type;

Полагаю, это хороший способ понять, как это работает на следующием практическом примере. Мы можем наблюдать шаг за шагом как makeIndexSequence<3> становится index\_sequence<0, 1, 2>.

* Мы видим, что makeIndexSequence<3> определена как typename indexSequenceHelper<3>::type (N равно трем).
* Компилятор сначала будет пытаться использовать обобщенную версию структуры indexSequenceHelper и он «попадет» в неё (N равно 3, а Next пустой). Этот шаблон наследует структуру indexSequenceHelper<2, 2>.
* Компилятор найдет, что структуре indexSequenceHelper<2, 2> снова соответствует этот же шаблон, который в свою очередь наследует структуру indexSequenceHelper<1, 1, 2> (N равно 2, а Next… состоит из единственного аргумента 2).
* Компилятор снова попадет в первый шаблон, который будет в свою очередь наследовать структуру indexSequenceHelper<0, 0, 1, 2> (N равно 1, а Next… состоит из двух аргументов 1, 2).
* Последняя в цепочке наследования структура indexSequenceHelper<0, 0, 1, 2> будет соответствовать частной специализации из-за 0-ого первого аргумента. Компилятор будет использовать второй шаблон для самой нижней структуры в получившейся цепочке иерархии. Эта структура будет иметь поле type, который будет инстанцирован значением indexSequence<0, 1, 2>.

NB: в библиотеке C++ структура integer\_sequence определена следующим образом.

*/// Class template integer\_sequence*

**template**<**typename** **\_Tp**, **\_Tp**... **\_Idx**>

**struct** **integer\_sequence**

{

**typedef** **\_Tp** **value\_type**;

**static** **constexpr** **size\_t** **size**() { **return** **sizeof**...(**\_Idx**); }

};

NB: структура содержит два члена: value\_type – тип аргументов пакета, size() – возвращает размер пакета (число аргументов в пакете).