std::iota

Defined in header <numeric>

`template< class ForwardIt, class T >`
`void iota( ForwardIt first, ForwardIt last, T value );`
	(since C++11)
	(until C++20)

`template< class ForwardIt, class T >`
`constexpr void iota( ForwardIt first, ForwardIt last, T value );`
	(since C++20)

Fills the range [first, last) with sequentially increasing values, starting with value and repetitively evaluating `++value`.

Equivalent operation:

```
*(d_first)   = value;
*(d_first+1) = ++value;
*(d_first+2) = ++value;
*(d_first+3) = ++value;
...
```

Parameters

first, last	-	the range of elements to fill with sequentially increasing values starting with value
value	-	initial value to store; the expression `++value` must be well-formed

Return value

(none)

Complexity

Exactly last - first increments and assignments.

Possible implementation

```
template<class ForwardIt, class T>
constexpr // since C++20
void iota(ForwardIt first, ForwardIt last, T value)
{
    while(first != last) {
        *first++ = value;
        ++value;
    }
}
```

Notes

The function is named after the integer function ⍳ from the programming language APL . It was one of the STL components (http://www.martinbroadhurst.com/stl/iota.html) that were not included in C++98, but made it into the standard library in C++11.

Example

The following example applies std::shuffle to a vector of std::lists' iterators. std::iota is used to populate containers.

Run this code

```
#include <algorithm>
#include <iomanip>
```

```
#include <iostream>
#include <list>
#include <numeric>
#include <random>
#include <vector>

class BigData // inefficient to copy
{
    int data[1024]; /* some raw data */
public:
    explicit BigData(int i = 0) { data[0] = i; /* ... */ }
    operator int () const { return data[0]; }
    BigData& operator=(int i) { data[0] = i; return *this; }
    /* ... */
};

int main()
{
    std::list<BigData> l(10);
    std::iota(l.begin(), l.end(), -4);

    std::vector<std::list<BigData>::iterator> v(l.size());
    std::iota(v.begin(), v.end(), l.begin());
    // Vector of iterators (to original data) is used to avoid expensive copying,
    // and because std::shuffle (below) cannot be applied to a std::list directly.

    std::shuffle(v.begin(), v.end(), std::mt19937{std::random_device{}()});

    std::cout << "Original contents of the list l:\t";
    for(auto const& n: l) std::cout << std::setw(2) << n << ' ';
    std::cout << '\n';

    std::cout << "Contents of l, viewed via shuffled v:\t";
    for(auto const i: v) std::cout << std::setw(2) << *i << ' ';
    std::cout << '\n';
}
```

Possible output:

```
Original contents of the list l:        -4 -3 -2 -1  0  1  2  3  4  5
Contents of l, viewed via shuffled v:   -1  5 -4  0  2  1  4 -2  3 -3
```

See also

ranges::iota_view (C++20) **views::iota**	a view consisting of a sequence generated by repeatedly incrementing an initial value (class template) (customization point object)
fill	copy-assigns the given value to every element in a range (function template)
ranges::fill (C++20)	assigns a range of elements a certain value (niebloid)
generate	assigns the results of successive function calls to every element in a range (function template)
ranges::generate (C++20)	saves the result of a function in a range (niebloid)
ranges::iota (C++23)	fills a range with successive increments of the starting value (niebloid)

Retrieved from "https://en.cppreference.com/mwiki/index.php?title=cpp/algorithm/iota&oldid=139004"