Fill and Generate Ranges

Next in the line of modifying algorithms, we have the 'fill' and 'generate' functions.

We can fill a range with `std::fill` and `std::fill_n`; we can generate new elements with `std::generate` and `std::generate_n`.

`fill`: Assigns `value` to each element in the range.

```
void fill(FwdIt first, FwdIt last, const T& val)
void fill(ExePol pol, FwdIt first, FwdIt last, const T& val)
```

`fill_n`: Starting at `first` and filling up to `n` elements, each element is assigned the value `value`.

```
OutIt fill_n(OutIt first, Size n, const T& val)
FwdIt fill_n(ExePol pol, FwdIt first, Size n, const T& val)
```

`generate`: The Generator generates a value `g` which is assigned to each element in the range.

```
void generate(FwdIt first, FwdIt last, Generator gen)
void generate(ExePol pol, FwdIt first, FwdIt last, Generator gen)
```

`generate_n`: The Generator generates a value `g` which is assigned to `n` values from the starting of the range at `first`.

```
OutIt generate_n(OutIt first, Size n, Generator gen)
FwdIt generate_n(ExePol pol, FwdIt first, Size n, Generator gen)
```

The algorithms expect the value `val` or a generator `gen` as an argument. `gen` has to be a function taking no arguments and returning the new value. The return value of the algorithms `std::fill_n` and `std::generate_n` is an output iterator, pointing to the last created element.

```cpp
#include <algorithm>
#include <iostream>
#include <list>
#include <vector>

int getNext(){
  static int next{0};
  return ++next;
}

int main(){

  std::cout << std::endl;

  std::vector<int> vec(20);
  std::fill(vec.begin(), vec.end(), 2011);
  for ( auto v: vec ) std::cout << v << " ";

  std::cout << std::endl;

  std::generate_n(vec.begin(), 15, getNext);
  for ( auto v: vec ) std::cout << v << " ";


  std::cout << "\n\n";

}
```

Fill and create ranges

In the next lesson, we’ll discuss how we can move data from one range to another.