Check Conditions on Ranges

C++17 contains several algorithms to check whether a value or values in a range fulfill our given condition. Let's look at these algorithms now.

The three functions `std::all_of`, `std::any_of` and `std::none_of` answer the question, if all, at least one or no element of a range satisfies the condition. The functions need as arguments input iterators and a unary predicate and return a boolean.

Checks if all elements of the range satisfy the condition:

```
bool all_of(InpIt first, InpIt last, UnPre pre)
bool all_of(ExePol pol, FwdIt first, FwdIt last, UnPre pre)
```

Checks if at least one element of the range satisfies the condition:

```
bool any_of(InpIt first, InpIt last, UnPre pre)
bool any_of(ExePol pol, FwdIt first, FwdIt last, UnPre pre)
```

Checks if no element of the range satisfies the condition:

```
bool none_of(InpIt first, InpIt last, UnPre pre)
bool none_of(ExePol pol, FwdIt first, FwdIt last, UnPre pre)
```

As promised, the example:

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

int main(){

  std::cout << std::boolalpha << std::endl;

  auto even= [](int i){ return i%2;};

  std::vector<int> myVec{1, 2, 3, 4, 5, 6, 7, 8, 9};
```

```
  std::cout << "std::any_of:\t" << std::any_of(myVec.begin(), myVec.end(), even) << std::endl
  std::cout << "std::all_of:\t" << std::all_of(myVec.begin(), myVec.end(), even) << std::endl
  std::cout << "std::none_of:\t" << std::none_of(myVec.begin(), myVec.end(), even) << std::er

 std::cout << std::endl;

}
```