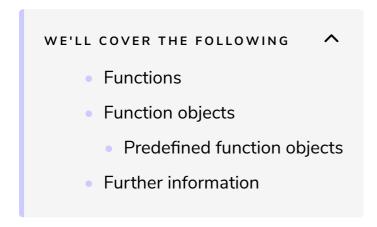
Functions and Function Objects

Let's take a look at the first two types of callables in C++.



Functions

Functions are the simplest callables. They can have, apart from static variables, no state. Because the definition of a function is often widely separated from its use or even in a different translation unit, the compiler has fewer opportunities to optimize the resulting code.

The code above takes squares for each of the values in the given set by using std::for_each algorithm.

Function objects

At first, don't call them functors. That's a *well-defined* term from the category theory.

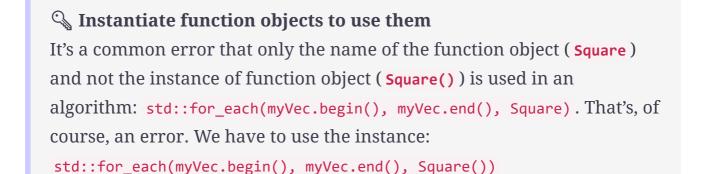
Function objects are objects that behave like functions. They achieve this due to their call operator being implemented. As function objects are objects, they can have attributes and therefore at state.

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

struct Square{
  void operator()(int& i){i= i*i;}
};

int main(){
  std::vector<int> myVec{1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
  std::for_each(myVec.begin(), myVec.end(), Square());
  for (auto v: myVec) std::cout << v << " "; // 1 4 9 16 25 36 49 64 81 100

  return 0;
}</pre>
```



Predefined function objects

C++ offers a bunch of predefined function objects. They need the header <functional>. These predefined function objects are very useful when changing the default behavior of a container. For example, the keys of the ordered associative containers are, by default, sorted with the predefined function object std::less. But we may want to use std::greater instead:

There are function objects in the Standard Template Library for arithmetic, logic and bitwise operations, and also for negation and comparison.

Function object for	Representative
Negation	<pre>std::negate<t>()</t></pre>
Arithmetic	<pre>std::plus<t>(), std::minus<t>()</t></t></pre>
	<pre>std::multiplies<t>(), std::divides<t>()</t></t></pre>
	<pre>std::modulus<t>()</t></pre>
Comparison	<pre>std::equal_to<t>(), std::not_equal_to<t>()</t></t></pre>
	<pre>std::less<t>(), std::greater<t>()</t></t></pre>
	<pre>std::less_equal<t>(), std::greater_equal<t>()</t></t></pre>
Logical	<pre>std::logical_not<t>()</t></pre>
	<pre>std::logical_and<t>(), std::logical_or<t>()</t></t></pre>
Bitwise	<pre>std::bit_and<t>(), std::bit_or<t></t></t></pre>
	std::bit_xor <t>()</t>

Further information

- functors
- Function objects

In the next lesson, we'll discuss a very important type of function: lambda functions.