- Example

In contrast to a function, a function object can have a state. The example in this lesson explains the point.

Operator overloading using parentheses

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

class SumMe{
public:

  SumMe(): sum(0){};

  void operator()(int x){
    sum += x;
  }

  int getSum() const {
    return sum;
  }
private:
  int sum;
};

int main(){

  std::cout << std::endl;

  std::vector<int> intVec = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

  SumMe sumMe = std::for_each(intVec.begin(), intVec.end(), SumMe());
  std::cout << "sumMe.getSum(): " << sumMe.getSum() << std::endl;

  std::cout << std::endl;

}
```

Explanation

- The `std::for_each` call in line 27 is a special algorithm of the Standard Template Library.

- It can return its callable. We invoke `std::for_each` with the function object `SumMe` and can, therefore, store the result of the function call directly in the function object.

- In line 28, we used the sum of all calls which is the state of the function object.

> **Note**: Lambda functions can also have a state.

In the next lesson, we'll solve an exercise to have more grip on Call operator.