Lambda Functions

Lambda functions provide us all the functionality we need, on the go. They are faster than user defined functions.

Lambda functions provide in-place functionality because the compiler gets a lot of insight and has therefore great optimization potential. Lambda functions can receive their arguments by value or by reference. They can capture their environment by value, by reference, and, with C++14, by `std::move`.

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

int main(){
  std::vector<int> myVec{1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
  std::for_each(myVec.begin(), myVec.end(), [](int& i){
    i= i*i;
    std::cout << i << " ";
  }); // 1 4 9 16 25 36 49 64 81 100

  return 0;
}
```

Lambda functions should be our first choice

If the functionality of our callable is short and self-explanatory, use a lambda function. A lambda function is, in general, faster and easier to understand.

In the next chapter, we'll discuss iterators and how they are used in C++ to traverse the data.