Removing Old functional Stuff

In this lessons we will look at some removeed functions.

Functions like `bind1st()`/`bind2nd()`/`mem_fun()`, … were introduced in the C++98-era and are not needed now as you can apply a lambda. What's more, the functions were not updated to handle perfect forwarding, `decltype` and other techniques from C++11. Thus it's best not to use them in modern code.

Removed functions:

- `unary_function()`/`pointer_to_unary_function()`
- `binary_function()`/`pointer_to_binary_function()`
- `bind1st()`/`binder1st`
- `bind2nd()`/`binder2nd`
- `ptr_fun()`
- `mem_fun()`
- `mem_fun_ref()`

For example to replace `bind1st`/`bind2nd` you can use lambdas or `std::bind` (available since C++11) or `std::bind_front` that should be available since C++20.

```
#include <functional>
#include <iostream>

int main() {
    auto onePlus = std::bind1st(std::plus<int>(), 1);
    std::cout << onePlus(10) << ',';
    auto minusOne = std::bind2nd(std::minus<int>(), 1);
    std::cout << minusOne(10) << '\n';

    // with hardcoded lambdas:
    auto lamOnePlus1 = [](int b) { return 1 + b; };
    std::cout << lamOnePlus1(10) << ',';
    auto lamMinusOne1 = [](int b) { return b - 1; };
    std::cout << lamMinusOne1(10) << '\n';

    // a capture with an initializer
    auto lamOnePlus = [a=1](int b) { return a + b; };
    std::cout << lamOnePlus(10) << ',';
    auto lamMinusOne = [a=1](int b) { return b - a; };

    std::cout << lamMinusOne(10) << '\n';

    // with bind:
    using namespace std::placeholders;
    auto onePlusBind = std::bind(std::plus<int>(), 1, _1);
    std::cout << onePlusBind(10) << ',';
    auto minusOneBind = std::bind(std::minus<int>(), _1, 1);
    std::cout << minusOneBind(10) << '\n';
}
```

> *Extra Info:* See more information in N4190.

The next lesson explores the deprecation of `std::iterator`.