C++ tricks

Any range printing (C++ 20)

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

template<class R>
void print(R range) {
    std::ranges::for_each(range, [](const auto& value) {
        std::cout << value << ' ';
    });
    std::cout << std::endl;
}

int main() {
    std::vector<int> vec = {1, 2, 3, 4, 5};
    print(vec);  // 1 2 3 4 5

    // algorithms are cool
    std::ranges::stable_partition(vec, [](const int& value) {
        return value % 2 == 0;
    });
    print(vec);  // 2 4 1 3 5

    return 0;
}
// Use g++ -std=c++20 test.cpp
```

Great tutorials

https://www.learncpp.com/

Inheritance and constructors

https://www.learncpp.com/cpp-tutorial/114-constructors-and-initialization-of-derived-classes/

Rule of three/five/zero

https://en.cppreference.com/w/cpp/language/rule_of_three
(In the cpp example, there is missing line: #include <utility>)

Details on what is generated when:
https://stackoverflow.com/questions/24342941/what-are-the-rules-for-automatic-generation-of-move-operations/24512883#24512883

Good reasons to avoid complex polymorphism and inheritance

C.67: A polymorphic class should suppress copying

General resources

Andrey Karpov's The Ultimate Question of Programming, Refactoring, and Everything
SCRAM coding standards

Using C code in C++

Use in header file:

```
#if defined(__cplusplus) /* If this is a C++ compiler, use C linkage */
extern "C" {
#endif

// ...

#ifdef __cplusplus /* If this is a C++ compiler, use C linkage */
}
#endif
```

More: https://stackoverflow.com/a/30526795/12118546

Excellent guide for the latest Intel SIMD AVX-512

https://colfaxresearch.com/knl-avx512/

Cool select, pool and epool Linux calls guide

https://jvns.ca/blog/2017/06/03/async-io-on-linux--select--poll--and-epoll/

Move semantics

https://docs.microsoft.com/en-us/cpp/cpp/move-constructors-and-move-assignment-operators-cpp?view=msvc-160
https://www.youtube.com/watch?v=St0MNEU5b0o

Lambdas

```
#include <iostream>

template<class T>
```

```
void print(T item) {
    std::cout << item << std::endl;
}

int main() {
    int i = 0;

    auto a = [=]() { return i; };
    auto b = [&]() { return i; };
    auto c = [i]() { return i; };
    auto d = [&i]() { return i; };

    ++i;

    print(a());  // 0
    print(b());  // 1
    print(c());  // 0
    print(d());  // 1

    return 0;
}
```