Алгоритмы стандартной библиотеки

- https://github.com/HappyCerberus/book-cpp-algorithms/

std::for_each, std::for_each_n

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

int main() {
    std::vector<int> data = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
    int sum1 = 0;
    for(auto el : data) {
        sum1 += el;
    }
    // sum1 == 45

    int sum2 = 0;
    std::for_each(data.begin(), data.end(), [&sum2](int el) {
        sum2 += el;
    });
    // sum2 == 45

    int sum3 = 0;
    std::for_each_n(data.begin(), data.size(), [&sum3](int el) {
        sum3 += el;
    });
    // sum3 == 45

    std::cout << sum1 << ' ' << sum2 << ' ' << sum3 << std::endl;
}
```

std::swap

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

int main() {
    std::string s1 = "First string";
    std::string s2 = "Second string";

    std::swap(s1, s2);
    std::cout << "s1: " << s1 << std::endl;
    std::cout << "s2: " << s2 << std::endl;

    // s1: Second string
    // s2: First string
}
```

std::sort, std::stable_sort, std::partial_sort

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

int main() {
    // O(nlogn), n = data1.size()
    std::vector<int> data1 = {9, 1, 8, 2, 7, 3, 6, 4, 5};
    std::sort(data1.begin(), data1.end());
    for (auto v : data1) {
        std::cout << v << " ";
    }
    std::cout << "\n";

    // O(nlogn), n = data2.size()
    std::list<int> data2 = {9, 1, 8, 2, 7, 3, 6, 4, 5};
    // std::sort(data.begin(), data.end()); // doesn't compile
    data2.sort();
    for (auto v : data2) {
        std::cout << v << " ";
    }
    std::cout << "\n";

    // O(nlogn), n = data3.size()
    std::vector<int> data3 = {9, 1, 8, 2, 7, 3, 6, 4, 5, 2};
    std::stable_sort(data3.begin(), data3.end());
    for (auto v : data3) {
        std::cout << v << " ";
    }
    std::cout << "\n";

    bool test1 = std::is_sorted(data1.begin(), data1.end());
    std::cout << test1 << std::endl; // true

    // O(nlogk), n = data4.size(), k = 3
    std::vector<int> data4{9, 8, 7, 6, 5, 4, 3, 2, 1};
    std::partial_sort(data4.begin(), data4.begin() + 3, data4.end());
    bool test2 = std::is_sorted(data4.begin(), data4.end());
    std::cout << test2 << std::endl; // false
}
```

std::partion, std::stable_partion

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

struct ExamResult {
    std::string student_name;
    int score;
};

std::vector<ExamResult> get_results() {
    return {
        {"Student 1", 100}, {"Student 2", 90}, {"Student 3", 15}, {"Student 4", 40},
        {"Student 5", 65}, {"Student 6", 75}, {"Student 7", 85}, {"Student 8", 30}
    };
}

int main() {
    std::vector<ExamResult> results = get_results();

    // O(n), n = results.size()
    auto pp = std::partition(results.begin(), results.end(),
        [threshold = 49](const auto& r) {
            return r.score >= threshold;
        });

    // process passing students
    for (auto it = results.begin(); it != pp; ++it) {
        std::cout << "[PASS] " << it->student_name << "\n";
    }
    std::cout << "\n";
    // process failed students
    for (auto it = pp; it != results.end(); ++it) {
        std::cout << "[FAIL] " << it->student_name << "\n";
    }
}
```

std::nth_element

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

int main() {
std::vector<int> data{9, 1, 8, 2, 7, 3, 6, 4, 5};
// O(n), n = data.size()
std::nth_element(data.begin(), data.begin() + 4, data.end());
std::cout << "data[4] == " << data[4] << "\n";

// O(n), n = data.size()
std::nth_element(data.begin(), data.begin() + 7, data.end(),
    std::greater<>());
std::cout << "data[7] == " << data[7] << "\n";
}
```

std::lower_bound, std::upper_bound

```cpp
#include <set>
#include <iostream>

int main() {
    std::multiset<int> data{1, 2, 3, 4, 5, 6, 6, 6, 7, 8, 9};

    auto lb = data.lower_bound(6);
    std::cout << "*lb == " << *lb << "\n";
    // std::distance(data.begin(), lb) == 5, *lb == 6

    auto ub = data.upper_bound(6);
    std::cout << "*ub == " << *ub << "\n";
    // std::distance(data.begin(), ub) == 8, *ub == 7

    std::vector<int> sorted_data(data.begin(), data.end());
    auto slb = std::lower_bound(sorted_data.begin(), sorted_data.end(), 6)
    std::cout << "*slb == " << *slb << "\n";
    // *ub == 6

    std::vector<int> sorted_data(data.begin(), data.end());
    auto sub = std::upper_bound(sorted_data.begin(), sorted_data.end(), 6)
    std::cout << "*sub == " << *sub << "\n";
    // *ub == 7
}
```

std::binary_search

```cpp
template<class ForwardIt, class T>
bool binary_search(ForwardIt first, ForwardIt last, const T& value)
{
    first = std::lower_bound(first, last, value);
    return (!(first == last) && !(value < *first));
}
```

std::binary_search

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

int main() {
    const auto haystack = {1, 3, 4, 5, 9};

    for (const auto needle : {1, 2, 3})
    {
        std::cout << "Searching for " << needle << '\n';
        if (std::binary_search(haystack.begin(), haystack.end(), needle))
            std::cout << "Found " << needle << '\n';
        else
            std::cout << "No dice!\n";
    }
}
```

std::fill, std::generate, std::fill_n, std::generate_n

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

int main() {
    std::vector<int> data(5, 0);
    for (auto v : data)
        std::cout << v << " ";
    std::cout << "\n";

    std::fill(data.begin(), data.end(), 11);
    for (auto v : data)
        std::cout << v << " ";
    std::cout << "\n";

    std::generate(data, []() { return 5; });
    for (auto v : data)
        std::cout << v << " ";
    std::cout << "\n";

    // iota-like
    std::generate(data, [i = 0]() mutable { return i++; });
    for (auto v : data)
        std::cout << v << " ";
    std::cout << "\n";
}
```

std::unique

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

int main() {
    std::vector<int> data{1, 1, 2, 2, 3, 4, 5, 6, 6, 6};

    auto it = std::unique(data.begin(), data.end());

    data.resize(std::distance(data.begin(), it));
    for (auto v : data) {
        std::cout << v << " ";
    }
    std::cout << "\n";
}
```

std::all_of, std::any_of, std::none_of

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

int main() {
    std::vector<int> data{-2, 0, 2, 4, 6, 8};

    bool all_even = std::all_of(data,
        [](int v) { return v % 2 == 0; });
    std::cout << std::boolalpha << "all_even == " << all_even << "\n";

    bool one_negative = std::any_of(data,
        [](int v) { return std::signbit(v); });
    std::cout << std::boolalpha << "one_negative == " << one_negative << "\n";

    bool none_odd = std::none_of(data,
        [](int v) { return v % 2 != 0; });
    std::cout << std::boolalpha << "none_odd == " << none_odd << "\n";
}
```

std::min, std::max

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

int main() {
    int x = 10, y = 20;
    int min = std::min(x, y);
    std::cout << "min == " << min << "\n";

    std::string hello = "hello", world = "world";
    std::string& universe = const_cast<std::string&>(std::max(hello, world));
    universe = "universe";

    std::string greeting = hello + " " + world;
    std::cout << "greeting == " << std::quoted(greeting) << "\n";

    int j = 20;
    auto& k = std::max(5, j);
    // IMPORTANT! only works because 5 < j
    // would produce dangling reference otherwise
    std::cout << "k == " << k << "\n";
}
```

std::set_difference, std::set_union, std::set_intersection

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

int main() {
    std::vector<int> data1{1, 2, 3, 4, 5, 6};
    std::vector<int> data2{3, 4, 5};

    std::vector<int> difference;
    std::set_difference(data1, data2,
        std::back_inserter(difference));

    for (auto v : difference) {
        std::cout << v << " ";
    }
    std::cout << "\n";

    std::vector<int> set_union;
    std::ranges::set_union(data1, data2,
    std::back_inserter(set_union));
    // set_union == { 1, 2, 3, 4, 5, 6 }

    data2 = std::vector<int>{2, 4, 6}
    std::vector<int> intersection;
    std::ranges::set_intersection(data1, data2,
    std::back_inserter(intersection));
    // intersection == {2, 4}
}
```

Streams. Manipulators

- https://www.studyplan.dev/pro-cpp/output-streams
- https://caiorss.github.io/C-Cpp-Notes/STL_Input_and_output.html

Buffered vs Unbuffered Streams

```cpp
#include <iostream>
#include <thread>

int main() {
    using namespace std::chrono_literals;
    std::cout << "Hello world!";
    std::this_thread::sleep_for(5s);
    std::cout << "\nDone!";
}
```

Flush

```cpp
#include <iostream>
#include <thread>

int main() {
    using namespace std::chrono_literals;
    std::cout << "Hello world!";
    std::cout.flush();
    std::this_thread::sleep_for(5s);
    std::cout << "\nDone!";
}
```

Flush

```cpp
#include <iostream>
#include <thread>

int main() {
    using namespace std::chrono_literals;
    std::cout << "Hello world!" << std::flush;
    std::this_thread::sleep_for(5s);
    std::cout << "\nDone!";
}
```

std::endl

```cpp
#include <iostream>
#include <thread>

int main() {
    using namespace std::chrono_literals;
    std::cout << "Hello world!" << std::endl;
    std::this_thread::sleep_for(5s);
    std::cout << "\nDone!";
}
```

Manipulators

```cpp
#include <iostream>

int main() {
    std::cout << std::hex;
    std::cout << "Hex: " << 255;
    std::cout << "\nStill Hex: " << 123;

    std::cout << std::dec;
    std::cout << "\n\nNow Decimal: " << 255;
    std::cout << "\nStill Decimal: " << 123;

    // Hex: ff
    // Still Hex: 7b
    // Now Decimal: 255
    // Still Decimal: 123
}
```

std::oct, std::dec, std::hex

```cpp
#include <iostream>

int main() {
    std::cout << std::oct << 255 << '\n';
    std::cout << std::dec << 255 << '\n';
    std::cout << std::hex << 255 << '\n';
    // 377
    // 255
    // ff
}
```

std::oct, std::dec, std::hex

```cpp
#include <iostream>

int main() {
    std::oct(std::cout);
    std::cout << 255 << '\n';

    std::dec(std::cout);
    std::cout << 255 << '\n';

    std::hex(std::cout);
    std::cout << 255 << '\n';

    // 377
    // 255
    // ff
}
```

iomanip

- https://en.cppreference.com/w/cpp/header/iomanip

std::setbase()

```cpp
#include <iomanip>
#include <iostream>

int main() {
    std::cout << std::setbase(8)  << 255 << '\n';
    std::cout << std::setbase(10) << 255 << '\n';
    std::cout << std::setbase(16) << 255 << '\n';
    // 377
    // 255
    // ff
}
```

std::boolalpha, std::noboolalpha

```
#include <iostream>

int main() {
    std::cout << std::boolalpha << true << ' '
                    << false << '\n';
    std::cout << std::noboolalpha << true << ' '
                    << false << '\n';
    // true false
    // 1 0
}
```

std::setprecision()

```cpp
#include <iomanip>
#include <iostream>

int main() {
    using std::cout, std::setprecision;
    float pi{3.141592};
    cout << setprecision(1) << pi << '\n';
    cout << setprecision(2) << pi << '\n';
    cout << setprecision(3) << pi << '\n';
    cout << setprecision(-1) << pi;
    // 3
    // 3.1
    // 3.14
    // 3.14159
}
```

precision

```cpp
#include <iostream>

int main() {
    std::cout.precision(3);
    std::cout << 3.1415;
    // 3.14
}
```

std::setw()

```cpp
#include <iomanip>
#include <iostream>

int main() {
    std::cout << std::setfill('-');
    std::cout << "Using std::setw(6):\n";
    std::cout << std::setw(6) << 1 << '\n';
    std::cout << std::setw(6) << 12 << '\n';
    std::cout << std::setw(6) << 123 << '\n';
    std::cout << std::setw(6) << 1234 << '\n';
    std::cout << std::setw(6) << 12345 << '\n';
    // -----1
    // ----12
    // ---123
    // --1234
    // -12345
}
```

put(), write()

```cpp
#include <iomanip>
#include <iostream>

int main() {
    std::cout.put('A');
    std::cout.put('\n');

    std::cout << std::setw(5);
    std::cout << "B\n";

    // This setw will be ignored
    std::cout << std::setw(10);
    std::cout.put('C');
    // A
    //     B
    // C
}
```

put(), write()

```cpp
#include <iomanip>
#include <iostream>

int main() {
    std::cout.write("Hello\n", 6);

    std::cout << std::setw(10);
    std::cout << "Hello\n";

    // setw will be ignored
    std::cout << std::setw(10);
    std::cout.write("Hello\n", 3);
    // Hello
    //     Hello
    // Hel
}
```

Standard Library IO Class Hierarchy

- https://caiorss.github.io/C-Cpp-Notes/STL_Input_and_output.html#org6360130

fstream

```cpp
#include <iostream>
#include <fstream>
#include <string>

int main () {
    std::ifstream myfile;
    myfile.open("shopping_list.txt");
    // std::ifstream myfile("shopping_list.txt");
    std::string mystring;
    if (myfile.is_open()) {
        myfile >> mystring;
        std::cout << mystring;
    }
}
```

openning modes

- `ios::app` Append mode. All output to that file is to be appended to the end.
- `ios::ate` Open a file for output and move the read/write control to the end of the file.
- `ios::in` Open a file for reading.
- `ios::out` Open a file for writing.
- `ios::trunc` If the file already exists, its contents will be truncated before opening the file.

https://itsourcecode.com/cplus-tutorial/learn-c-files-and-streams-with-best-program-examples/

seekg

```
// position to the nth byte of fileObject (assumes ios::beg)
fileObject.seekg( n );

// position n bytes forward in fileObject
fileObject.seekg( n, ios::cur );

// position n bytes back from end of fileObject
fileObject.seekg( n, ios::end );

// position at end of fileObject
fileObject.seekg( 0, ios::end );
```