Numerical Algorithms

CSE 232 - Dr. Josh Nahum

Reading:

Section 17.1 - Section 17.4

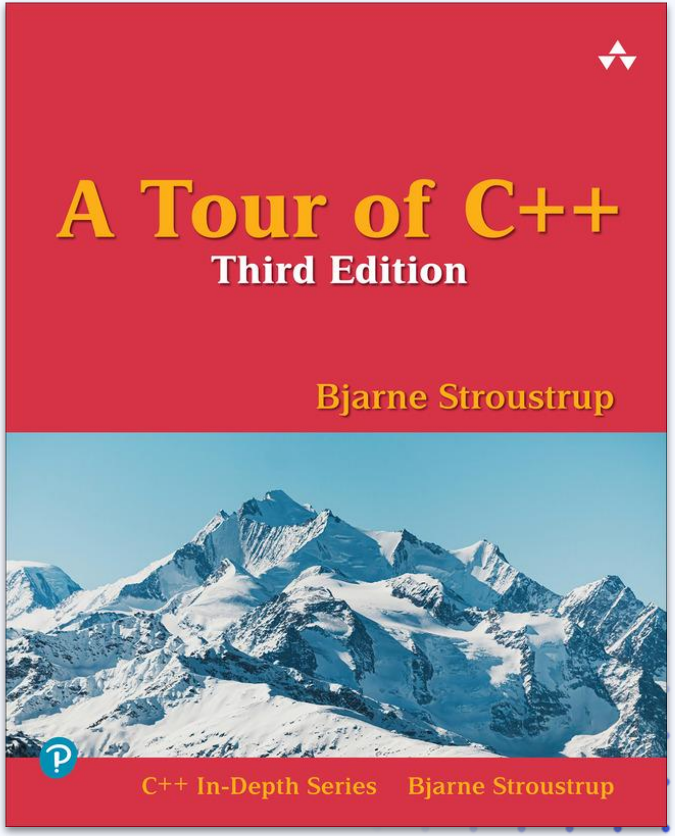

Table of contents

00
Math Functions

01
Accumulate

02
Execution Policies

03
Complex Numbers

00

Math Functions

`std::` required?

The all the functions provided in <cmath> are in the std namespace. However, some standard library implementations also pollute the global namespace with these function declarations too. Don't rely on this, use the std:: namespace specification!

Sometimes only some of the functions are added to the global namespace, and thus a "`using namespace std;`" declaration can actually affect which overloaded function is called. NOT GOOD!

01

Accumulate

Calculate sum

```
std::vector<double> temps {
  98, 97, 98.6, 101};
double sum = std::accumulate(
  temps.begin(), temps.end(), double{0});
// 394.6
```

The most common use case for std::accumulate is to make a tally/sum.
Note that the type of the third argument determines the type that is returned and used internally.

Calculate product

```cpp
std::vector<int> factors{2, 4, 29};
double product = std::accumulate(
  factors.begin(), factors.end(), int{1},
  [](int a, int b){return a * b;});
// 232
```

An optional 4th argument can be given to replace the use of operator+.

Note that to calculate the product, the initial value should be 1, not 0.

Concatenate

```cpp
std::string CommaSeparate(std::string const& left,
    std::string const& right) {
  return left + "," + right;
}
// ...
std::vector<std::string> names{"Mal", "Kira", "Dax"};
std::string line = std::accumulate(
  names.begin(), names.end(),
  std::string{"Josh"}, CommaSeparate);
  // Josh,Mal,Kira,Dax
```

Accumulate can work on non-numeric types like string.

02

Execution Policies

Parallel Algorithms

Many algorithms have an overload with an Execution Policy that can be used to indicate that the algorithm should use parallelism to run faster.

Parallelism can allow code to run faster, but it has startup costs and synchronizations costs, some operations may introduce data races, and debugging is more difficult.

When to use an execution policy other than seq (short for sequential, the default) is beyond the scope of this course.

03

Complex Numbers

Accumulate and Complex

```
std::vector<std::complex<int>> vec =
    {{{2,3}, {4,5}, {10, -30}}};
int imag_sum = std::accumulate(
  vec.begin(), vec.end(), int{0},
  [](int x, std::complex<int>y) {
    return x + y.imag();
  });
// -22
```

Accumulate can be used on vectors of complex numbers.

Here we sum the imaginary components of the complex numbers.

Note that the third argument doesn't have to match the elements of the container.

Attribution

Please ask questions via Piazza

Dr. Joshua Nahum
www.nahum.us
EB 3504

CREDITS: This presentation template was created by **Slidesgo**, and includes icons by **Flaticon**, and infographics & images by **Freepik**