Initializer Lists

CSE 232 - Dr. Josh Nahum

Reading:

Section 5.2.3

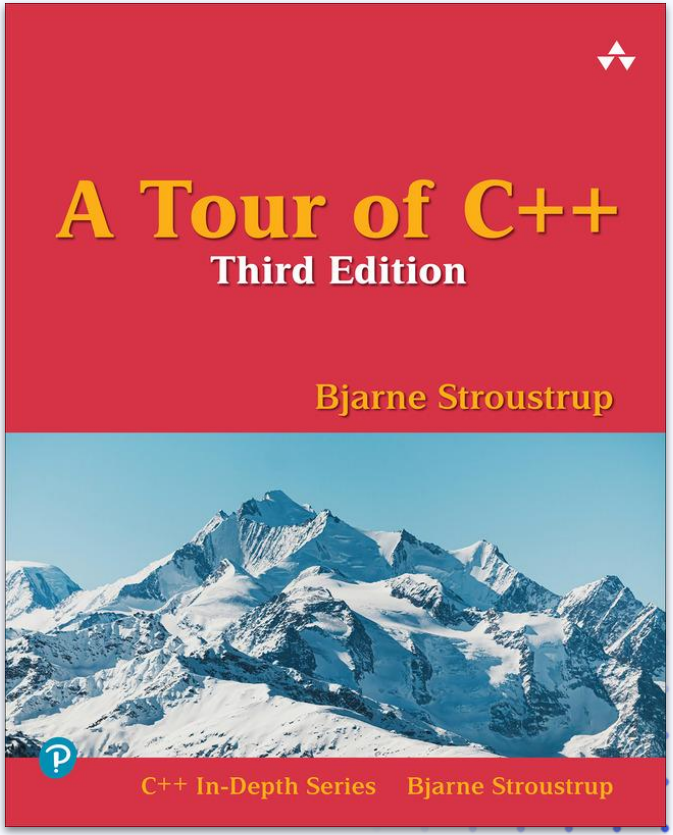

Table of contents

00

Streams

read Function

```
Vector read(istream & is) {
  Vector v;
  for (double d; is >> d; ) {
    v.push_back(d);
  }
  return v;
}
```

Two facts to remember:

- `is >> d` is an expression that is true as long as the stream is not in an error state.
- Streams can't be copied, but you can pass references (or pointers) to them to functions.

01

Copy

Initializer-list Constructor

```
Vector::Vector(std::initializer_list<double> lst)
  :elem{new double[lst.size()]}, sz{static_cast<int>(lst.size())} {
  copy(lst.begin(), lst.end(), elem);
}
```

`std::copy` is a function from the `<algorithm>` library. It takes three arguments:

1. An iterator to the first element in a container to be copied
2. An iterator to one past the last element in a container to be copied
3. An iterator to the first element in a container to be overwritten by the copy

An example without using untaught material follows.

Initializer-list Constructor

```
Vector::Vector(std::initializer_list<double> lst)
  :elem{new double[list.size()]}, sz{static_cast<int>(lst.size())} {
  for (int i{0}; i < sz; ++i) {
    elem[i] = lst[i];
  }
}
```

Unfortunately arrays (like elem) and std::initializer_lists (like lst) don't support the `at` method, so we have to use the less safe **`operator[]`**.

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**