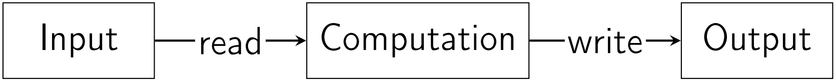
The Standard Template Library



- Dealing with data as sequence of elements

The Standard Template Library

$$\begin{tikzcd}
\boxed{\text{Input}} \arrow[r, "\text{read}"] & \boxed{\text{Computation}} \arrow[r, "\text{write}"] & \boxed{\text{Output}}
\end{tikzcd}$$

- Sort the words in a dictionary in order
- Find the highest temperature this month
- Count occurrences of a word on the web.

The Standard Template Library

$$\begin{tikzcd}\boxed{\text{Input}} \arrow[r, "\text{read}"] & \boxed{\text{Computation}} \arrow[r, "\text{write}"] & \boxed{\text{Output}}\end{tikzcd}$$

- Containers: `std::vector`, `std::list`
- Iterators: `begin`, `end`
- Algorithms: `std::sort`, `std::find`

Containers

```
#include <vector>
std::vector v{1, 2, 3, 4, 5};
for(auto i : v) {
  std::cout << i << std::endl;
}
// Equivalent
for (auto i = v.begin(); i != v.end(); ++i) {
  std::cout << *i << std::endl;
}
v.push_back(6);
```

- Contiguous
- Variable size
- Fast access, slow insertion and deletion

Containers

```cpp
#include <list>
std::list l{1, 2, 3, 4, 5};
for(auto i : l) {
  std::cout << i << std::endl;
}
l.push_back(6);
// v[2] = 7; // Error
```

- Not contiguous
- Slow access, fast insertion and deletion

Containers

```
#include <deque>
std::deque d{1, 2, 3, 4, 5};
for(auto i : d) {
  std::cout << i << std::endl;
}
d.push_back(6);
d.push_front(0);
v[2] = 7; // Ok
```

- Not contiguous
- Variable size
- Fast access, slow insertion and deletion except at beginning/end

Associative containers

```
#include <map>
std::map<std::string, int> m;
m["one"] = 1;
m["two"] = 2;
m["three"] = 3;
for(auto i : m) {
  std::cout << i.first << " " << i.second <<
   ↪  std::endl;
 } // one 1 two 2 three 3
```

- Key-value pairs. Only unique keys.
- O(log(N)) access, insertion and removal.
- `std::unordered_map` for O(1) access.

Associative containers

```
#include <set>
std::set<int> s;
s.insert(1);
s.insert(1);
s.insert(3);
for(auto i : s) {
  std::cout << i << std::endl;
 } // 1 3
```

- Unique keys
- O(log(N)) access, insertion and removal
- `std::unordered_set` for O(1) access.