COMP6771
Advanced C++ Programming

2.1 – Standard Library Overview

Software Libraries

- Most of us are quite familiar with libraries in software. For example, when programming in C, we frequently use <stdio.h> and <stdlib.h>.
- Being an effective programmer often consists of the effective use of libraries. In some ways, this becomes more important than being a genius at writing code from scratch. Don't reinvent the wheel!
- It is essential to know what facilities your favourite language’s Standard Library provides before using 3rd Party libraries!

C++ Standard Library

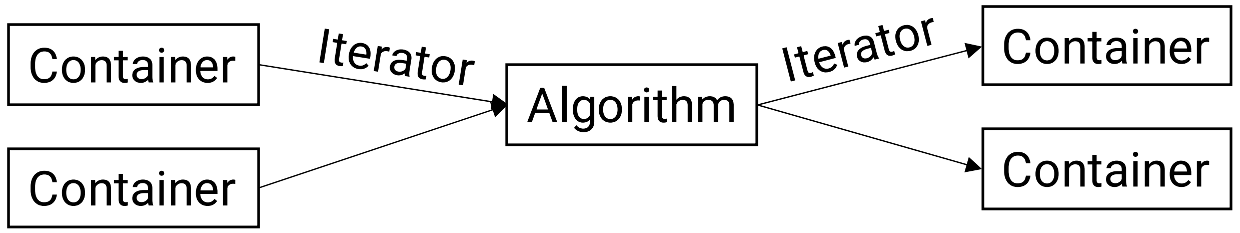
C++ offers a wealth of ready-to-use functions and types in its Standard Library:

- The original C Standard Library
- A containers library
- An iterators library
- An algorithms library
- A new string library
- A memory management library
- A new random numbers library
- Ranges
- String
- And many more!

The containers, iterators, and algorithms libraries are collectively known as the Standard Template Library.
We will be focusing on these.

STL: Standard Template Library

- STL is an architecture and design philosophy for managing generic and abstract collections of data with algorithms.
- All components of the STL are templates.
- Containers store data, but don't know about algorithms.
- Iterators are an API to access items within a container in a particular order, agnostic of the container used.
  - Each container has its own iterator types.
- Algorithms manipulate values referenced by iterators, but don't know about containers.



Why STL?

- STL offers an assortment of containers
- STL publicizes the time and storage complexity of its containers
- STL containers grow and shrink in size automatically
- STL provides built-in algorithms for processing containers
- STL provides iterators that make the containers and algorithms flexible and efficient.
- STL is extensible which means that users can add new containers and new algorithms such that:
  - STL algorithms can process STL containers as well as user defined containers
  - User defined algorithms can process STL containers as well user defined containers

Introductory Example

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

bool is_even(int n) { return n % 2 == 0; }

int main() {
  auto v1 = std::vector<int>{1, 2, 3, 4, 5, 6, 7, 8};
  auto v2 = std::vector(4); // makes space for 4 elems

  // use standard algorithm to copy only the elements that satisfy “is_even”.
  // begin() and end() return iterators
  std::copy_if(v1.begin(), v1.end(), v2.begin(), is_even);

  for (int elem : v2) { // the ranged for-loop also secretly uses iterators!
      std::cout << elem << std::endl;
  } // prints 2, 4, 6, 8
}
```

Feedback (stop recording)