The C++ STL

The C++ Standard Template Library is a major component of the c++ standard library it is a large collection of general-purpose generic classes, functions, and iterators

- Generic containers take the element type as a parameters that take an iterator and perform an interesting operation on the elements in that range
- different kinds of iterators that can navigate thru the containers
- algorithms that take an iterator ...

Design Philosophy

A collection of useful, efficient, type-safe, generic containers

- a container has almost no understanding of the element type (excepting ordered containers can be sorted via a less than operator)
- each container should define its own iterators
- container methods use a static dispatch, no virtual

A collection of useful, efficient, generic algorithms that operate on its iterators

- generic => algorithm knows nothing about the structure its operating on, apart from the fact that it can be traversed by an iterator, and knows almost nothing about the elements in the structure
- define container methods only when the generic algorithms are unsuitable or much less efficient

Some Facts about STL

The STL assumes little about the contained elements

- the element type must allow copying and assignment
- this is legal: $vector < vector < string >> v$

For ordered collections, the element type must support operator< or you can provide a special functor(function-object) of your own.

The STL assumes value semantics for its contained elements; objects are copied to and from containers more often than you might think (when a vector doubles in size for example).

No Inheritance in STL

Guy who made it did not like inheritance. For the sake of true generality templates are used to achieve a more flexible kind of polymorphism (duck typing)

The algorithms are deigned so that almost any algorithm can be used with any STL container or any other data structure that supports the idea of iterators.

the container are just different enough that code reuse isn't really practical. No container methods are virtual in the interests of efficiency.

Polymorphic Containers

Suppse we want to model a graphical Scene that has an ordered list of Figures(Square, Circle, Rectangle, etc)

- recal Figure is an abstract base calss so it cant be instantiated
- Scene will have a textual Caption plus the list of figures which we can implement using a vector

To draw this we can contain the data in:

- vectorFigure