**Vectors**

1. **The Containers Library**

**std::vector**

Vector is a sequence of elements that you can access by index. Vectors can be thought of as arrays that can grow (and shrink) in length while your program is running.

**Recall**: The **type of the vector** (i.e., what data type is stored inside) cannot be changed after the declaration. However, the vector can be double even though he values entered are int.

**Creating a vector**

vector *<base\_type>* name;

**Initialize a Vector**

Creating and initializing vector <double> location = {42.651443, -73.749302};

Presizing (setting the size) vector <double> location (2);

Setting vector full of values vector <double> location (50, 2); // {2, 2, … 2}

🡪 The vector will have default element values of 0.0 (default value for double); If the vector base type is a class, the default constructor is used for initialization

*NOTE* You can use v[i] to change the value of the ith element. However, you cannot initialize the ith element using v[i] if vector is not presized; you can only change an element that has already been given some value.

We can also iterate over vector elements using range-based for-loop

**Members**

.size() Number of elements in the vector

.capacity() Number of elements for which it currently has memory allocated (>= size)

**Adding or Removing elements**

Adding new element to the end: .push\_back() 🡪 Only way to initialize vector element if not presized

Removing element from the back: .pop\_back() **NOTE that pop in C++ has no return value**

**Shrinking size and capacity**

*(since capacity is automatically doubled when extra is needed, which may affect efficiency)*

Adding capacity to the vector manually: .reserve (<int>)

Resizing (either cutting or adding 0-value elements): .resize(<int>)

**Copy** - Deep copy is implemented

**Destructors** - Sufficient memory management has been done (no need to delete [] vect)