**Vector Containers**

#include <iostream>

#include <vector>

using *std*::*cout*;

using *std*::*endl*;

using *std*::*vector*;

// Vector - An STL (Standard Template Library) container which is an expandable array, expands in batches

// Vectors are great at working with very last element, adding a new element, taking off an element

// Not great for removing items from the middle of it's container.

int main()

{

//vector<int> myVector; // {}

*vector*<int> myVector(6, 0); // {0,0,0,0,0}

// resizes array [adds one more to back] and passes the value of the parameter, and it assigns to the new last element in the array

myVector.*push\_back*(10); // {0,0,0,0,0,10}

// removes the last index and value of that index from the array

myVector.*pop\_back*(); // {0,0,0,0,0}

// Since Vector is an object, it has a size function built into it's functionality that returns the length of the vector array

for (int i = 0; i < myVector.*size*(); i++)

{

myVector[i] = i; // {0, 1, 2, 3, 4}

*cout* << myVector[i] << " " << *endl*;

}

// access an element exactly like you would an array. Grabs the index which starts at zero.

*cout* << "The element at index 3 (or element 4) is : " << myVector[3] << *endl*; // 3

// gets last element

*cout* << "Last element is : " << myVector.*back*() << *endl*;

// gets first element

*cout* << "First element is : " << myVector.*front*() << *endl*;

// erase elements from your vector (same for other containers) - use iterators {0, 1, 2, 3, 4, 5} - myVector.begin() - .begin() is an iterator

// for erasing a single element

myVector.*erase*(myVector.*begin*() + 3); // erases element at index 3 {0, 1, 2, 4, 5}

// for erasing a range of elements (index start, index stop - 1)

myVector.*erase*(myVector.*begin*() + 2, myVector.*begin*() + 4); // erases elements starting at index 2 up to but not including element index 4. {0, 1, 4, 5}

*vector* <int> newVector;

for (int i = 0; i < 32; i++)

{

newVector.*push\_back*(i);

*cout* << "#" << i << " size: " << newVector.*size*();

*cout* << ", capacity: " << newVector.*capacity*() << *endl*;

}

return 0;

}