Vectors

CSCE1040

Quick into: Vectors

Vectors are same as dynamic arrays with the ability to resize itself automatically when an element is inserted or deleted, with their storage being handled automatically by the container. Vector elements are placed in contiguous storage so that they can be accessed and traversed using iterators. In vectors, data is inserted at the end.

Vectors in C++

- A ***vector*** is an ordered list of items of a given data type.
- A vector was added to C++ as a safer and more powerful form of arrays.
- To use vector you have to add: **#include <vector>**
- In a vector access, the number in **.at()** parentheses is called the ***index*** of the corresponding element. The first vector element is at index 0.
- A vector's **size()** function returns the number of vector elements.

How to declare vector

```
vector<dataType> vectorName(numElements);
```

Vector Initialization

- A vector's elements are automatically initialized to 0s during the vector declaration.
- All of a vector's elements may be initialized to another single value. Ex: vector<int> myVector(3, -1); creates a vector named myVector with three elements, each with value -1.
- A programmer may initialize each vector element with different values by specifying the initial values in braces {} separated by commas. Ex: vector<int> carSales = {5, 7, 11};

Commonly-used vector member functions

at()	`at(size_type n)`
	
	Accesses element n.
		`teamNums.at(3) = 99;  // Assigns 99 to element 3`
		`x = teamNums.at(3);  // Assigns element 3's value 99 to x`

size()	`size_type size() const;`
	
	Returns vector's size.
		`if (teamNums.size() > 0) {  // Size is 5 so condition is true`
		`  ...`
		`}`

empty()	`bool empty() const;`
	
	Returns true if size is 0.
		`if (teamNums.empty()) {  // Size is 5 so condition is false`
		`  ...`
		`}`

clear()	Removes all elements. Vector size becomes 0.	`teamNums.clear();  // Vector now has no elements`
		`cout << teamNums.size();  // Prints 0`
		`teamNums.at(3) = 88;  // Error; element 3 does not exist`

push_back()	`void push_back(const T& x);`
	
	Copies x to new element at vector's end, increasing size by 1. Parameter is pass by reference to avoid making local copy, but const to make clear not changed.
		`// Assume vector is empty`
		`teamNums.push_back(77);  // Vector is: 77`
		`teamNums.push_back(88);  // Vector is: 77, 88`
		`cout << teamNums.size(); // Prints 2`

erase()	`iterator erase (iteratorPosition);`
	
	Removes element from position. Elements from higher positions are shifted back to fill gap. Vector size decrements.
		`// Assume vector is 77, 33, 88`
		`teamNums.erase(teamNums.begin() + 1); // Now 77, 88`
		`// (Strange position indication explained below)`

insert()	`iterator insert(iteratorPosition, const T& x);` Copies x to element at position. Items at that position and higher are shifted over to make room. Vector size increments.	`// Assume vector is 77, 88` `teamNums.insert(teamNums.begin() + 1, 33); // Now 77, 33, 88`

Vectors in our Lab4

In ShoppingCart.h we will have to include the following into our declaration of a class ShoppingCart:

vector<ItemToPurchase> cartItems;

After that we will update our vector in ShoppingCart.cpp