CS 225

Data Structures

Feb. 8 — Stacks and Queues Wade Fagen-Ulmschneider, Craig Zilles

List.h

Array Implementation

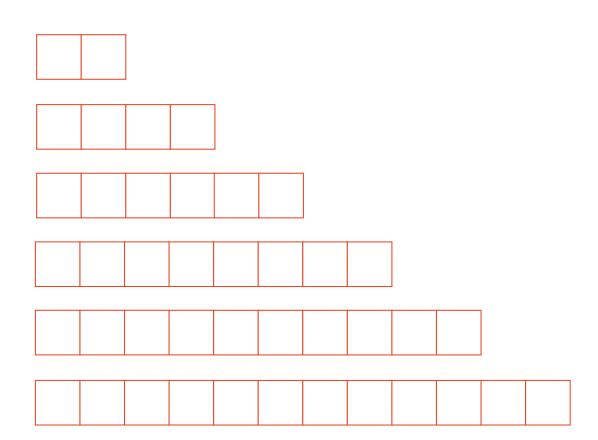
С	S	2	2	5
[0]	[1]	[2]	[3]	[4]

Array Implementation

insertAtFront:

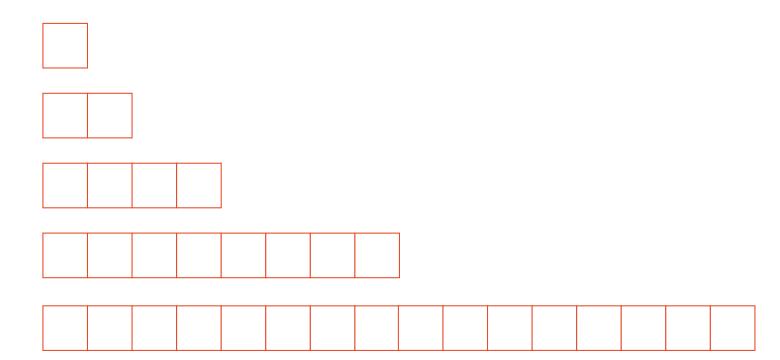


Resize Strategy: +2 elements every time



Resize Strategy: +2 elements every time

Resize Strategy: x2 elements every time

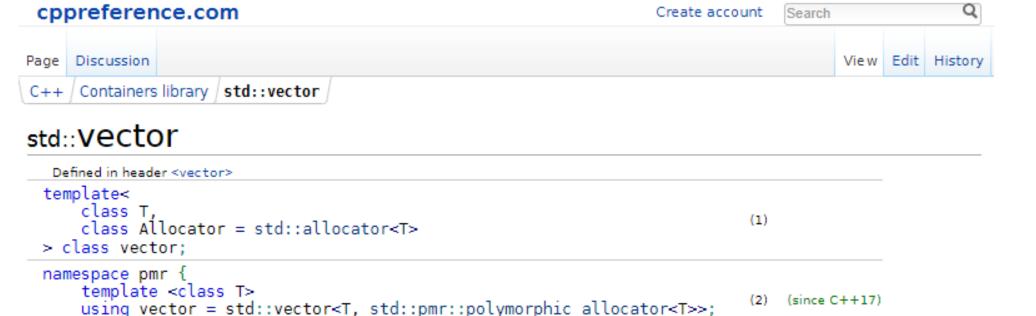


Resize Strategy: x2 elements every time

Array Implementation

	Singly Linked List	Array
Insert/Remove at front		
Insert at given element		
Remove at given element		
Insert at arbitrary location		
Remove at arbitrary location		

std::vector



- 1) std::vector is a sequence container that encapsulates dynamic size arrays.
- 2) std::pmr::vector is an alias template that uses a polymorphic allocator

The elements are stored contiguously, which means that elements can be accessed not only through iterators, but also using offsets to regular pointers to elements. This means that a pointer to an element (since C++03) of a vector may be passed to any function that expects a pointer to an element of an array.

The storage of the vector is handled automatically, being expanded and contracted as needed. Vectors usually occupy more space than static arrays, because more memory is allocated to handle future growth. This way a vector does not need to reallocate each time an element is inserted, but only when the additional memory is exhausted. The total amount of allocated memory can be queried using capacity() function. Extra memory can be returned to the system via a call to shrink to fit(). (since C++11)

Element access	
at	access specified element with bounds checking (public member function)
operator[]	access specified element (public member function)
front	access the first element (public member function)
back	access the last element (public member function)
data(C++11)	direct access to the underlying array (public member function)

Modifiers			
clear	clears the contents (public member function)		
insert	inserts elements (public member function)		
emplace(c++11)	constructs element in-place (public member function)		
erase	erases elements (public member function)		
push_back	adds an element to the end (public member function)		
emplace_back(c++11)	constructs an element in-place at the end (public member function)		
pop_back	removes the last element (public member function)		
resize	changes the number of elements stored (public member function)		
swap	swaps the contents (public member function)		

Capacity	resize	chang (public	
empty	checks whether the container is empty (public member function)	swap sv	
size	returns the number of elements (public member function)		
max_size	returns the maximum possible number of elements (public member function)		
reserve	reserves storage (public member function)		
capacity	returns the number of elements that can be held in currently allocated storage (public member function)		
shrink_to_fit(c++11)	reduces memory usage by freeing unused mem (public member function)	nory	

Stack ADT

Queue ADT