Tutorial 3
Collection

Jonathan Kristian Tjandra (USTF)
SDS, 122040024

Outlines

- Review: vector, stack, queue, map, set
- Code examples
- Debugging demonstrations
- Q&A

Vector

The implementation of vector is based on array on C++, but there are differences

- Vector is **dynamic**, array is **static**
- Vector's size can **shrink and grow** depending on needs, while array's size is **fixed** at initialization
- Vector can be directly copied to other vector
- Vector can be treated as an array

Member functions

(constructor)	Construct vector (public member function)
(destructor)	Vector destructor (public member function)
operator=	Assign content (public member function)

Iterators:

begin	Return iterator to beginning (public member function)
end	Return iterator to end (public member function)
rbegin	Return reverse iterator to reverse beginning (public member function)
rend	Return reverse iterator to reverse end (public member function)
cbegin C++11	Return const_iterator to beginning (public member function)
cend C++11	Return const_iterator to end (public member function)
crbegin C++11	Return const_reverse_iterator to reverse beginning (public member function)
crend C++11	Return const_reverse_iterator to reverse end (public member function)

Capacity:

size	Return size (public member function)
max_size	Return maximum size (public member function)
resize	Change size (public member function)
capacity	Return size of allocated storage capacity (public member function)
empty	Test whether vector is empty (public member function)
reserve	Request a change in capacity (public member function)
shrink_to_fit C++11	Shrink to fit (public member function)

Element access:

operator[]	Access element (public member function)
at	Access element (public member function)
front	Access first element (public member function)
back	Access last element (public member function)
data C++11	Access data (public member function)

Modifiers:

assign	Assign vector content (public member function)
push_back	Add element at the end (public member function)
pop_back	Delete last element (public member function)
insert	Insert elements (public member function)
erase	Erase elements (public member function)
swap	Swap content (public member function)
clear	Clear content (public member function)
emplace C++11	Construct and insert element (public member function)
emplace_back C++11	Construct and insert element at the end (public member function)

Allocator:

get_allocator	Get allocator (public member function)

Non-member function overloads

relational operators	Relational operators for vector (function template)
swap	Exchange contents of vectors (function template)

Template specializations

vector<bool>	Vector of bool (class template specialization)

To learn more, visit cplusplus.com/reference

Methods in the `Vector<type>` Class

`vec.size()` Returns the number of elements in the vector.		
`vec.isEmpty()` Returns **`true`** if the vector is empty.		
`vec.get(i)` Returns the i^{th} element of the vector.	*or*	**`vec[i]`**
`vec.set(i, value)` Sets the i^{th} element of the vector to **`value`**.	*or*	**`vec[i] = value;`**
`vec.add(value)` Adds a new element to the end of the vector.	*or*	**`vec += value;`**

Methods in the `Vector<type>` Class

Constructor		
Vector()	O(1)	Initializes a new empty vector.
Vector(n, value)	O(N)	Initializes a new vector storing n copies of the given value.

Methods		
add(value)	O(1)	Adds a new value to the end of this vector.
clear()	O(1)	Removes all elements from this vector.
equals(v)	O(N)	Returns **true** if the two vectors contain the same elements in the same order.
get(index)	O(1)	Returns the element at the specified index in this vector.
insert(index, value)	O(N)	Inserts the element into this vector before the specified index.
isEmpty()	O(1)	Returns **true** if this vector contains no elements.
mapAll(fn)	O(N)	Calls the specified function on each element of the vector in ascending index order.
remove(index)	O(N)	Removes the element at the specified index from this vector.
set(index, value)	O(1)	Replaces the element at the specified index in this vector with a new value.
size()	O(1)	Returns the number of elements in this vector.
subList(start, length)	O(N)	Returns a new vector containing elements from a sub-range of this vector.
toString()	O(N)	Converts the vector to a printable string representation.

Operators		
v[index]	O(1)	Overloads **[ ]** to select elements from this vector.
v1 + v2	O(N)	Concatenates two vectors.
v1 += v2;	O(N)	Adds all of the elements from **v2** to **v1**.
v += value;	O(1)	Adds the single specified value to **v**.
v += a, b, c;	O(1)	Adds multiple individual values to **v**.
v1 == v1	O(N)	Returns **true** if **v1** and **v2** contain the same elements.
v1 != v2	O(N)	Returns **true** if **v1** and **v2** are different.
ostream << v	O(N)	Outputs the contents of the vector to the given output stream.
istream >> v	O(N)	Reads the contents of the given input stream into the vector.

Methods in the STL `vector<type>` Class

fx Member functions	
(constructor)	Construct vector (public
(destructor)	Vector destructor (public
operator=	Assign content (public m

Iterators:	
begin	Return iterator to begin
end	Return iterator to end (
rbegin	Return reverse iterator
rend	Return reverse iterator
cbegin C++11	Return const_iterator to
cend C++11	Return const_iterator to
crbegin C++11	Return const_reverse_it
crend C++11	Return const_reverse_it

Capacity:	
size	Return size (public mem
max_size	Return maximum size (
resize	Change size (public mem
capacity	Return size of allocated
empty	Test whether vector is e
reserve	Request a change in cap
shrink_to_fit C++11	Shrink to fit (public mem

Element access:	
operator[]	Access element (public member function)
at	Access element (public member function)
front	Access first element (public member function)
back	Access last element (public member function)
data C++11	Access data (public member function)

Modifiers:	
assign	Assign vector content (public member function)
push_back	Add element at the end (public member function)
pop_back	Delete last element (public member function)
insert	Insert elements (public member function)
erase	Erase elements (public member function)
swap	Swap content (public member function)
clear	Clear content (public member function)
emplace C++11	Construct and insert element (public member function)
emplace_back C++11	Construct and insert element at the end (public member function)

Allocator:	
get_allocator	Get allocator (public member function)

fx Non-member function overloads	
relational operators	Relational operators for vector (function template)
swap	Exchange contents of vectors (function template)

Template specializations	
vector<bool>	Vector of bool (class template specialization)

Different!

Stack

Follows the **LIFO (Last in, First Out)** property

- **Use cases**: Bracket matching, postfix evaluation, finding smallest integer after removing n digits
- Unlike Stanford Library's Stack, the STL Stack's pop method **only deletes** the last element and does not return it. Use **top** first to access the element

fx Member functions

(constructor)	Construct stack (public member function)
empty	Test whether container is empty (public member function)
size	Return size (public member function)
top	Access next element (public member function)
push	Insert element (public member function)
emplace C++11	Construct and insert element (public member function)
pop	Remove top element (public member function)
swap C++11	Swap contents (public member function)

Queue

Follows the **FIFO (First In, First Out)** property

fx Member functions

(constructor)	Construct queue (public member function)
empty	Test whether container is empty (public member function)
size	Return size (public member function)
front	Access next element (public member function)
back	Access last element (public member function)
push	Insert element (public member function)
emplace C++11	Construct and insert element (public member function)
pop	Remove next element (public member function)
swap C++11	Swap contents (public member function)

Map

Contains **keys** and **values** that makes up **key-pairs**

- Datatypes between keys and values can be different
- The keys in std::map are **sorted** (for more efficient searching)
- For unsorted keys use std::unordered_map
- Values in maps are accessed using keys, not indexes like in arrays and vectors
- Each key will only appear once in a map, meaning that if we insert a key that already exists, the operation will be cancelled

fx Member functions

(constructor)	Construct map (public member function)
(destructor)	Map destructor (public member function)
operator=	Copy container content (public member function)

Iterators:

begin	Return iterator to beginning (public member function)
end	Return iterator to end (public member function)
rbegin	Return reverse iterator to reverse beginning (public member function)
rend	Return reverse iterator to reverse end (public member function)
cbegin C++11	Return const_iterator to beginning (public member function)
cend C++11	Return const_iterator to end (public member function)
crbegin C++11	Return const_reverse_iterator to reverse beginning (public member function)
crend C++11	Return const_reverse_iterator to reverse end (public member function)

Capacity:

empty	Test whether container is empty (public member function)
size	Return container size (public member function)
max_size	Return maximum size (public member function)

Element access:

operator[]	Access element (public member function)
at C++11	Access element (public member function)

Modifiers:

insert	Insert elements (public member function)
erase	Erase elements (public member function)
swap	Swap content (public member function)
clear	Clear content (public member function)
emplace C++11	Construct and insert element (public member function)
emplace_hint C++11	Construct and insert element with hint (public member function)

Observers:

key_comp	Return key comparison object (public member function)
value_comp	Return value comparison object (public member function)

Operations:

find	Get iterator to element (public member function)
count	Count elements with a specific key (public member function)
lower_bound	Return iterator to lower bound (public member function)
upper_bound	Return iterator to upper bound (public member function)
equal_range	Get range of equal elements (public member function)

Allocator:

get_allocator	Get allocator (public member function)

Set

Very similar to set (usually discussed in set theories) in mathematics

- Each element in a set is **not sorted**
- Like maps, each value in a set only appears once
- Can be implemented using a map

Member functions

(constructor)	Construct set (public member function)
(destructor)	Set destructor (public member function)
operator=	Copy container content (public member function)

Iterators:

begin	Return iterator to beginning (public member function)
end	Return iterator to end (public member function)
rbegin	Return reverse iterator to reverse beginning (public member function)
rend	Return reverse iterator to reverse end (public member function)
cbegin C++11	Return const_iterator to beginning (public member function)
cend C++11	Return const_iterator to end (public member function)
crbegin C++11	Return const_reverse_iterator to reverse beginning (public member function)
crend C++11	Return const_reverse_iterator to reverse end (public member function)

Capacity:

empty	Test whether container is empty (public member function)
size	Return container size (public member function)
max_size	Return maximum size (public member function)

Modifiers:

insert	Insert element (public member function)
erase	Erase elements (public member function)
swap	Swap content (public member function)
clear	Clear content (public member function)
emplace C++11	Construct and insert element (public member function)
emplace_hint C++11	Construct and insert element with hint (public member function)

Observers:

key_comp	Return comparison object (public member function)
value_comp	Return comparison object (public member function)

Operations:

find	Get iterator to element (public member function)
count	Count elements with a specific value (public member function)
lower_bound	Return iterator to lower bound (public member function)
upper_bound	Return iterator to upper bound (public member function)
equal_range	Get range of equal elements (public member function)

Allocator:

get_allocator	Get allocator (public member function)

Iterating over a collection

Strategy 1: To go through each element in order

- Very easy to implement on collections that used index to access their elements
- Not very clear on others that doesn’t use index, e.g., maps and sets

Strategy 2: To use iterators

- More modern way and better way to approach this problem
- Needs knowledge of pointers and addresses to understand with more detail

Iterating over a collection

Fortunately, C++ has simplified iterators by enabling ranged-based for loops

For example:

```
vector<int> v;
v.push(5);
v.push(6);
for(int value : v){
        cout << value << endl;
}
```

Example Code 1: Tic Tac Toe

Description: Given a string of Tic Tac Toe game state (length of 9), we need to check whether a player wins the game or not

Example input & output:

Input:
“XXX O O “
“X OXO XO “
“OX XOX O“

Output:
X wins
X wins
O wins

Note: The quotation marks is not included in the input. It is there to indicate the start and end of the input string

Example Code 2: Bracket Checking

Description: Given a string, we need to check whether the brackets in said string is properly matched or not

Example input & output:

Input:
{s=2*(a[2]+3);x=(1+(2));}
(a[2] + b[3)
(){{}}

Output:
Brackets are properly nested
Brackets are incorrect
Brackets are properly nested

Example Code 3: Symbol Table

- Write a C++ program that declares such a symbol table and then reads in command lines from the user, which must be in one of the following forms:
  - A simple assignment statement of the form *var* **=** *number*.
  - A variable alone on a line, which is a request to display its value.
  - The command **`quit`**, which exits from the program.
  - The command **`list`**, which lists all the variables.

Debugging Demonstration: Postfix Evaluation

Debugging Demo: Postfix Evaluation

Description: Given a string of a postfix expression, we need to evaluate the final result of such expression

Example input & output:

Input:	**Output:**
2 1 + 3 *	9
4 13 5 / +	6.6
10 6 9 3 + -11 * / * 17 + 5 +	21.5455

Q&A