# ALL STL FUNCTIONS

vector functions are push\_back,capacity(the size it can take ),size(for the current size),max\_size(fully how many bites in can take),

empty(),front(for last element),back(last element),at(the index value we want to know), vectors can be printed like array v[i],

as well as it can be printed as iterator through begin() & end()

begin() – Returns an iterator pointing to the first element in the vector

end() – Returns an iterator pointing to the theoretical element that follows the last element in the vector

rbegin() – Returns a reverse iterator pointing to the last element in the vector (reverse beginning). It moves from last to first element

rend() – Returns a reverse iterator pointing to the theoretical element preceding the first element in the vector (considered as reverse end)

cbegin() – Returns a constant iterator pointing to the first element in the vector.

cend() – Returns a constant iterator pointing to the theoretical element that follows the last element in the vector.

crbegin() – Returns a constant reverse iterator pointing to the last element in the vector (reverse beginning). It moves from last to first element

crend() – Returns a constant reverse iterator pointing to the theoretical element preceding the first element in the vector (considered as reverse end)

size() – Returns the number of elements in the vector.

max\_size() – Returns the maximum number of elements that the vector can hold.

capacity() – Returns the size of the storage space currently allocated to the vector expressed as number of elements.

resize(n) – Resizes the container so that it contains ‘n’ elements.

empty() – Returns whether the container is empty.

shrink\_to\_fit() – Reduces the capacity of the container to fit its size and destroys all elements beyond the capacity.

reserve() – Requests that the vector capacity be at least enough to contain n elements.

reference operator [g] – Returns a reference to the element at position ‘g’ in the vector

at(g) – Returns a reference to the element at position ‘g’ in the vector

front() – Returns a reference to the first element in the vector

back() – Returns a reference to the last element in the vector

data() – Returns a direct pointer to the memory array used internally by the vector to store its owned elements.

assign() – It assigns new value to the vector elements by replacing old ones

push\_back() – It push the elements into a vector from the back

pop\_back() – It is used to pop or remove elements from a vector from the back.

insert() – It inserts new elements before the element at the specified position

erase() – It is used to remove elements from a container from the specified position or range.

swap() – It is used to swap the contents of one vector with another vector of same type. Sizes may differ.

clear() – It is used to remove all the elements of the vector container

emplace() – It extends the container by inserting new element at position

emplace\_back() – It is used to insert a new element into the vector container, the new element is added to the end of the vector

------------------------------------------------------------------------------------------------

list functions are push\_back(pushing elements from back line),push\_front(pushing elements from front line),pop\_back(rremove element from the back),pop\_front(remove element from the front),reverse(to reverse the list),

sort(to sort a list),remove(element which have to be removed should be inthe bracet),clear(to clear the whole list),

size(to know the currnt size of the list),it can be printed as iterator through begin() & end()

front() – Returns the value of the first element in the list.

back() – Returns the value of the last element in the list .

push\_front(g) – Adds a new element ‘g’ at the beginning of the list .

push\_back(g) – Adds a new element ‘g’ at the end of the list.

pop\_front() – Removes the first element of the list, and reduces size of the list by 1.

pop\_back() – Removes the last element of the list, and reduces size of the list by 1

list::begin() and list::end() in C++ STL– begin() function returns an iterator pointing to the first element of the list

end()– end() function returns an iterator pointing to the theoretical last element which follows the last element.

list rbegin() and rend() function in C++ STL– rbegin() returns a reverse iterator which points to the last element of the list. rend() returns a reverse iterator which points to the position before the beginning of the list.

list cbegin() and cend() function in C++ STL– cbegin() returns a constant random access iterator which points to the beginning of the list. cend() returns a constant random access iterator which points to the end of the list.

list crbegin() and crend() function in C++ STL– crbegin() returns a constant reverse iterator which points to the last element of the list i.e reversed beginning of container. crend() returns a constant reverse iterator which points to the theoretical element preceding the first element in the list i.e. the reverse end of the list.

empty() – Returns whether the list is empty(1) or not(0).

insert() – Inserts new elements in the list before the element at a specified position.

erase() – Removes a single element or a range of elements from the list.

assign() – Assigns new elements to list by replacing current elements and resizes the list.

remove() – Removes all the elements from the list, which are equal to given element.

list::remove\_if() in C++ STL– Used to remove all the values from the list that correspond true to the predicate or condition given as parameter to the function.

reverse() – Reverses the list.

size() – Returns the number of elements in the list.

list resize()function in C++ STL– Used to resize a list container.

sort() – Sorts the list in increasing order.

list max\_size() function in C++ STL– Returns the maximum number of elements a list container can hold.

list unique() in C++ STL– Removes all duplicate consecutive elements from the list.

list::emplace\_front() and list::emplace\_back() in C++ STL– emplace\_front() function is used to insert a new element into the list container, the new element is added to the beginning of the list. emplace\_back() function is used to insert a new element into the list container, the new element is added to the end of the list.

list::clear() in C++ STL– clear() function is used to remove all the elements of the list container, thus making it size 0.

list::operator= in C++ STL– This operator is used to assign new contents to the container by replacing the existing contents.

list::swap() in C++ STL– This function is used to swap the contents of one list with another list of same type and size.

list splice() function in C++ STL– Used to transfer elements from one list to another.

list merge() function in C++ STL– Merges two sorted lists into one

list emplace() function in C++ STL– Extends list by inserting new element at a given position.

----------------------------------------------------------------------------------------------------

map functions are insert(pair<data type(key), data type(value)>(key,value)),erase(remove value by erasing the key),find(enter the key then the value will be deleted),erase(map\_name.begin(), map\_name.find(the key till which to remove)),

max\_size,

begin() – Returns an iterator to the first element in the map

end() – Returns an iterator to the theoretical element that follows last element in the map

size() – Returns the number of elements in the map

max\_size() – Returns the maximum number of elements that the map can hold

empty() – Returns whether the map is empty

pair insert(keyvalue, mapvalue) – Adds a new element to the map

erase(iterator position) – Removes the element at the position pointed by the iterator

erase(const g)– Removes the key value ‘g’ from the map

clear() – Removes all the elements from the map

map insert() in C++ STL– Insert elements with a particular key in the map container. .

map count() function in C++ STL– Returns the number of matches to element with key value ‘g’ in the map.

map equal\_range() in C++ STL– Returns an iterator of pairs. The pair refers to the bounds of a range that includes all the elements in the container which have a key equivalent to k.

map erase() function in C++ STL– Used to erase element from the container.

map rend() function in C++ STL– Returns a reverse iterator pointing to the theoretical element right before the first key-value pair in the map(which is considered its reverse end).

map rbegin() function in C++ STL– Returns a reverse iterator which points to the last element of the map.

map find() function in C++ STL– Returns an iterator to the element with key value ‘g’ in the map if found, else returns the iterator to end.

map crbegin() and crend() function in C++ STL– crbegin() returns a constant reverse iterator referring to the last element in the map container. crend() returns a constant reverse iterator pointing to the theoretical element before the first element in the map.

map cbegin() and cend() function in C++ STL– cbegin() returns a constant iterator referring to the first element in the map container. cend() returns a constant iterator pointing to the theoretical element that follows last element in the multimap.

map emplace() in C++ STL– Inserts the key and its element in the map container.

map max\_size() in C++ STL– Returns the maximum number of elements a map container can hold.

map upper\_bound() function in C++ STL– Returns an iterator to the first element that is equivalent to mapped value with key value ‘g’ or definitely will go after the element with key value ‘g’ in the map

map operator= in C++ STL– Assigns contents of a container to a different container, replacing its current content.

map lower\_bound() function in C++ STL– Returns an iterator to the first element that is equivalent to mapped value with key value ‘g’ or definitely will not go before the element with key value ‘g’ in the map.

map emplace\_hint() function in C++ STL– Inserts the key and its element in the map container with a given hint.

map value\_comp() in C++ STL– Returns the object that determines how the elements in the map are ordered (‘<' by default).

map key\_comp() function in C++ STL– Returns the object that determines how the elements in the map are ordered (‘<' by default).

map::size() in C++ STL– Returns the number of elements in the map.

map::empty() in C++ STL– Returns whether the map is empty.

map::begin() and end() in C++ STL– begin() returns an iterator to the first element in the map. end() returns an iterator to the theoretical element that follows last element in the map

map::operator[] in C++ STL– This operator is used to reference the element present at position given inside the operator.

map::clear() in C++ STL– Removes all the elements from the map.

map::at() and map::swap() in C++ STL– at() function is used to return the reference to the element associated with the key k. swap() function is used to exchange the contents of two maps but the maps must be of same type, although sizes may differ.

--------------------------------------------------------------

stack functions are push(to enter element from the front),top(to print the first element),pop(to remove first element),

empty() – Returns whether the stack is empty – Time Complexity : O(1)

size() – Returns the size of the stack – Time Complexity : O(1)

top() – Returns a reference to the top most element of the stack – Time Complexity : O(1)

push(g) – Adds the element ‘g’ at the top of the stack – Time Complexity : O(1)

pop() – Deletes the top most element of the stack – Time Complexity : O(1)

stack::top() in C++ STL

stack::empty() and stack::size() in C++ STL

stack::push() and stack::pop() in C++ STL

stack::swap() in C++ STL

stack::emplace() in C++ STL

Recent Articles on C++ Stack

---------------------------------------------------------------------

queue functions are push(to enter element from the back),size(to print th size ),front(to prin tthe first element),back(to print the last element),

pop(to remove element from the front),

empty() – Returns whether the queue is empty.

size() – Returns the size of the queue.

queue::swap() in C++ STL: Exchange the contents of two queues but the queues must be of same type, although sizes may differ.

queue::emplace() in C++ STL: Insert a new element into the queue container, the new element is added to the end of the queue.

queue::front() and queue::back() in C++ STL– front() function returns a reference to the first element of the queue. back() function returns a reference to the last element of the queue.

push(g) and pop() – push() function adds the element ‘g’ at the end of the queue. pop() function deletes the first element of the queue.

---------------------------------------------------------------------------

set

begin() – Returns an iterator to the first element in the set.

end() – Returns an iterator to the theoretical element that follows last element in the set.

size() – Returns the number of elements in the set.

max\_size() – Returns the maximum number of elements that the set can hold.

empty() – Returns whether the set is empty.

begin() – Returns an iterator to the first element in the set.

end() – Returns an iterator to the theoretical element that follows last element in the set.

rbegin()– Returns a reverse iterator pointing to the last element in the container.

rend()– Returns a reverse iterator pointing to the theoretical element right before the first element in the set container.

crbegin()– Returns a constant iterator pointing to the last element in the container.

crend() – Returns a constant iterator pointing to the position just before the first element in the container.

cbegin()– Returns a constant iterator pointing to the first element in the container.

cend() – Returns a constant iterator pointing to the position past the last element in the container.

size() – Returns the number of elements in the set.

max\_size() – Returns the maximum number of elements that the set can hold.

empty() – Returns whether the set is empty.

insert(const g) – Adds a new element ‘g’ to the set.

iterator insert (iterator position, const g) – Adds a new element ‘g’ at the position pointed by iterator.

erase(iterator position) – Removes the element at the position pointed by the iterator.

erase(const g)– Removes the value ‘g’ from the set.

clear() – Removes all the elements from the set.

key\_comp() / value\_comp() – Returns the object that determines how the elements in the set are ordered (‘<‘ by default).

find(const g) – Returns an iterator to the element ‘g’ in the set if found, else returns the iterator to end.

count(const g) – Returns 1 or 0 based on the element ‘g’ is present in the set or not.

lower\_bound(const g) – Returns an iterator to the first element that is equivalent to ‘g’ or definitely will not go before the element ‘g’ in the set.

upper\_bound(const g) – Returns an iterator to the first element that is equivalent to ‘g’ or definitely will go after the element ‘g’ in the set.

equal\_range()– The function returns an iterator of pairs. (key\_comp). The pair refers to the range that includes all the elements in the container which have a key equivalent to k.

emplace()– This function is used to insert a new element into the set container, only if the element to be inserted is unique and does not already exists in the set.

emplace\_hint()– Returns an iterator pointing to the position where the insertion is done. If the element passed in the parameter already exists, then it returns an iterator pointing to the position where the existing element is.

swap()– This function is used to exchange the contents of two sets but the sets must be of same type, although sizes may differ.

operator= – The ‘=’ is an operator in C++ STL which copies (or moves) a set to another set and set::operator= is the corresponding operator function.

get\_allocator()– Returns the copy of the allocator object associated with the set.