Listas(list):

iterator_t insert(iterator_t p , elem_t x): inserta el elemento x en la posición p, devolviendo una posición q al elemento insertado,ya que las posiciones de p en adelante quedan inválidas.
iterator_t erase(iterator_t p): borra el elemento de la posición p, devolviendo una posición q al elemento que previamente estaba, ya que p en adelante quedan inválidas.
elem_t &retrieve(iterator_t p): devuelve el elemento que hay en la posición p.
iterator_t next(iterator_t p): devuelve la posición siguiente a p.
iterator_t begin(): devuelve la primera posición de la lista
iterator_t end(): devuelve la última posición de la lista
push_back(elem_t x): agrega al final de la lista el elemento x
void unique( ): elimina los elementos duplicados.
void sort( ): ordena la lista.
Void reverse( ): invierte la lista.

pilas(stack):

elem_t top(): devuelve el elemento del tope de la pila
pop(): elimina el elemento del tope
push(elem_t x): inserta en el tope de la pila el elemento x
clear(): vacía la pila
int size(): devuelve el tamaño de la pila
bool empty(): retorna verdadero si la pila esta vacía

cola(queue):

clear(): vacía la cola
bool empty(): retorna verdadero si la cola esta vacía
elem_t front(): devuelve el elemento del frente de la cola
pop(): elimina el elemento del frente de la cola
push(elem_t x): inserta el elemento x a lo último de la cola
int size(): devuelve el tamaño
clear(): vacía la cola

vector:

end():devuelve referencia al último elemento
begin():devuelve referencia al primer elemento
clear(): vacía el vector
empty():verdadero si el vector esta vacío
erase():elimina un elemento específico
insert():inserta un elemento x en la posición p del vector
push_back(elem_t x): agrega al final del vector el elemento x
int size(): nos dice el tamaño del vector
pop_back(): elimina el último elemento
void resize(int n): cambia el tamaño del vector por el tamaño n.

map

int size(): nos da el tamaño del contenedor
iterator_t find(k): dada la clave k,devuelve un iterador a su par.si k no tiene asignado un valor, entonces devuelve end()
insert(pair<tipo,tipo>(k,val)): asigna a la clave k el valor val.si k ya tenia algo, entonces lo reemplaza.devuelve un iterador al par correspondiente.(no pisa el valor)

val=M[k] o range_t& retrieve(k): recupera el valor asignado val a la clave k. Ahora si k no tiene asignado nada, entonces crea y devuelve un valor por defecto.
p->first : devuelve la clave de la asignación apuntada por p.
p->second : devuelve el valor de la asignación apuntada por p.
erase(iterator_t p): elimina la asignación apuntada por p, donde p puede ser el iterador a la clave o el valor.
erase(iterator_t inicio , iterator_t fin): borra el rango de valores
erase(k): borra el contenido de la posición donde esta la clave k.
Void clear(): elimina todo

arboles(tree):

iterator_t p.lchild( ): dada una posición p, retorna el hijo mas izquierdo.
iterator_t p.right( ): dada una posición p,retorna la posición del hermano derecho.
iterator_t end( ):retorna la posición final no referenciable del árbol.
iterator_t begin( ):retorna la posición del comienzo del árbol.,es decir, la raíz.
elem_t &retrieve(iterator_t p):retorna una referencia al valor que hay en el nodo p.
iterator_t insert(iterator_t p , elem_t x): inserta un nuevo nodo en la posición p con el elemento x. p puede ser referenciable o no, y devuelve la posición al nuevo elemento insertado.
iterator_t erase(iterator p): elimina el nodo o posición p, y todo el subárbol que contiene.
Void clear( ): elimina todos los elementos del árbol.
iterator_t splice(iterator_t to, interator_t from): elimina todo el subárbol del nodo from y lo inserta en el nodo to. Tanto to como from no deben tener relación de antecesor o descendiente y pueden estar en diferentes árboles.
iterator_t find(elem_t x): devuelve un iterador al nodo donde se encuentra el elemento x (función de interfaz avanzada)
iterator_t splice(iterator_t to, iterator_t from):elimina todo el subárbol del nodo from y lo inserta en el nodo(desreferenciable o no) to. To y from no deben tener relación de antecesor o descendiente y pueden estar en diferentes árboles. El nodo to a donde vamos a pasar todo el subárbol se convierte en hermano del subárbol.

Arboles binarios(btree)

iterator end( ):retorna la posición final no referenciable del árbol.
iterator begin( ):retorna la posición del comienzo del árbol.,es decir, la raíz.
Iterator left( ): devuelve una referencia al hijo izquierdo del nodo actual.
Iterator right( ): devuelve un iterador al hijo derecho del nodo actual
iterator insert(iterator p , elem_t x): inserta un nuevo nodo en la posición p con el elemento x. p puede ser referenciable o no, y devuelve la posición al nuevo elemento insertado.
iterator erase(iterator p): elimina el nodo o posición p, y todo el subárbol que contiene.
Void clear( ): elimina todos los elementos del árbol.
iterator splice(iterator to, iterator from):elimina todo el subárbol del nodo from y lo inserta en el nodo(desreferenciable o no) to. To y from no deben tener relación de antecesor o descendiente y pueden estar en diferentes árboles. El nodo to a donde vamos a pasar todo el subárbol se convierte en hermano del subárbol.

Conjuntos(set)

iterator insert(elem x): inserta el elemento x, devolviendo un iterator(iterador al elemento insertado).si no lo inserta devuelve end().
iterator insert(iterator p, elem x): inserta el elemento x en la posición referenciada p, y devuelve un iterador al elemento insertado.
void erase(iterator p): elimina el elemento referenciado por el iterador p.
int erase(elem x): retorna el número de elementos eliminados x efectivamente

iterator begin( ): retorna iterador al comienzo del conjunto.
iterator end( ): retorna iterador al final del conjunto.
void clear( ): vacía el conjunto.
Iterator find(elem x): busca el elemento x y retorna un iterador al elemento si lo encuentra, del contrario retorna end( ).
bool empty( ): retorna verdadero si el conjunto esta vacío.
Int size( ): retorna el tamaño del conjunto.

algoritmos de la STL

elem_t min(elem_t a, elem_t b): devuelve el mínimo entre los valores a y b.
elem_t max(elem_t a,elem_t b):devuelve el valor más grande entre a y b.
elem_t min_element( ):devuelve el mínimo elemento en una secuencia.
elem_t max_element( ):devuelve el máximo elemento en una secuencia.
void swap( ):intercambia 2 valores.
bool equal( ): detecta si dos secuencias son iguales.
iterator find(iterator first, iterator last, elem_t valor): busca la primera ocurrencia de un valor en la secuencias.
f**or_each(iterator first, iterator last, function F):** aplica una operación a cada elemento.
count_if(iterator first, iterator last, predicado): cuenta el número de elementos que satisfacen un predicado.