Guía rápida de STL de C++

Notación

FWIter	Forward Iterator	Iterador de avance
BDIter	Bidirectional Iterator	Iterador bidireccional
INIter	Input Iterator	Iterador de lectura
OUTIter	Output Iterator	Iterador de escritura
RNDIter	Random Access Iterator	Iterador de acceso aleatorio (vector y deque)
PredFunc	Predicate	Función lógica que recibe un parámetro y retorna un valor booleano
BinPredFunc	Binary Predicate	Función lógica que recibe dos parámetros y retorna un valor booleano
Func	Function	Función que retorna void
UNFunc	Unary Function	Función que recibe un parámetro T y retorna un valor de tipo T
BINFunc	Binary Function	Función que recibe dos parámetros T y retorna un valor de tipo T
GenFunc	Generator	Función que no recibe parámetros y devuelve un valor del tipo T. Se utiliza para generar números según algún patrón

Contenedores: operaciones comunes

Operación	**Descripción**
X::size()	Devuelve la cantidad de elementos que tiene el contenedor como un entero sin signo
X::max_size()	Devuelve el tamaño máximo que puede alcanzar el contenedor antes de requerir más memoria
X::empty()	Retorna true si el contenedor no tiene elementos
X::swap(X & x)	Intercambia el contenido del contenedor con el que se recibe como parámetro
X::clear()	Elimina todos los elementos del contenedor
v == w, v != w	Supóngase que existen dos contenedores del mismo tipo: *v* y *w*. Todas las comparaciones se hacen lexicográficamente y retornan un valor booleano.
v < w, v > w	
v <= w, v >= w	

Contenedores lineales: operaciones comunes

S::push_back(T & x)	Inserta un elemento al final de la estructura
S::pop_back()	Elimina un elemento del final de la estructura
S::front()	Devuelve una referencia al primer elemento de la lista
S::back()	Devuelve una referencia al último elemento de la lista

#include <vector>

operator[](int i)	Devuelve una referencia al elemento que está en la posición i (el primer elemento tiene índice cero)
resize(int i)	Da un nuevo tamaño al contenedor especificado por cant

#include <deque>

vector	Todas las funcionalidades de vector
push_front(T & x)	Inserta el elemento x al inicio del contenedor
pop_front()	Elimina el primer elemento de la secuencia

#include <list>

push_front(T & x)	Inserta el elemento x al inicio del contenedor
pop_front()	Elimina el primer elemento de la lista
remove(T & valor)	Elimina todas las apariciones de valor en la secuencia
remove_if(PredFunc pred)	Elimina todos los elementos que satisfacen la función pred
unique()	Elimina los elementos iguales consecutivos
unique(BinPredFunc binpred)	Elimina los elementos consecutivos que satisfacen el predicado
reverse()	Invierte la lista
sort()	Ordena de menor a mayor la lista

Iteradores

Forward	==, !=, *, ++
Bidirectional	Forward, --
Random access	Bidirectional, <, >, <=, >=, +, -, +=, -=

#include <algorithm>

return	Algoritmo	Parámetros	Descripción
int	count	FWIter primero, FWIter ultimo, T val	Devuelve la cantidad de apariciones de val
int	count_if	FWIter primero, FWIter ultimo, PredFunc pred	Devuelve la cantidad de elementos que satisfacen la función predicado
INIter	find	INIter primero, INIter ultimo, T val	Devuelve un iterador al primer elemento igual a val, o ultimo si no existe

INIter	find_if	INIter primero, INIter ultimo, PredFunc pred	Devuelve un iterador al primer elemento que satisface la función
bool	equal	INIter1 primero1, INIter1 ultimo1, INIter2 primero2	Devuelve true si ambos tienen los mismos elementos
FWIter1	search	FWIter1 primero1, FWIter1 ultimo1, FWIter2 primero2, FWIter2 ultimo2	Localiza la aparición del segundo contenedor en el primero
FWIter	min_element	FWIter primero, FWIter ultimo	Devuelve un iterador al menor elemento
FWIter	max_element	FWIter primero, FWIter ultimo	Devuelve un iterador al mayor elemento
OUTIter	copy	INIter primero1, INIter ultimo1, OUTIter primero2	Copia el contenido del primero en el segundo
void	swap	T & p, T & q	Intercambia los valores de p y q
Func	for_each	INIter primero, INIter ultimo, Func f	Aplica la función f a cada elemento del contenedor
OUTIter	transform	INIter primero, INIter ultimo, OUTIter result, UNFunc f	Aplica f a cada elemento del contenedor y guarda el resultado en result
OUTIter	transform	INIter1 primero1, INIter1 ultimo1, INIter2 primero2, OUTIter result, BINFunc f	Aplica f a ambos contenedores y guarda el resultado en result
void	replace	FWIter primero, FWIter ultimo, T & valor_viejo, T & valor_nuevo	Reemplaza todos los valores viejos por los nuevos
void	fill	FWIter primero, FWIter ultimo, T & val	Llena el contenedor con val
void	generate	FWIter prinero, FWIter ultimo, GenFunc f	Llena el contenedor con los valores que devuelve la función f
FWIter	remove	FWIter primero, FWIter ultimo, T & val	Elimina todas las apariciones de val
FWIter	unique	FWIter primero, FWIter ultimo	Elimina los elementos contiguos repetidos
void	reverse	BDIter primero, BDIter ultimo	Invierte el orden de la lista
void	rotate	FWIter primero, FWIter centro, FWIter ultimo	Rota la lista hacia la derecha hasta que primero == centro
void	random_shuffle	RNDIter primero, RNDIter ultimo	Reordena los elementos al azar
void	sort	RNDIter primero, RNDIter ultimo	Ordena los elementos de menor a mayor

#include <numeric>

T	accumulate	INIter primero, INIter ultimo, T val_ini	Da como resultado val_ini mas la sumatoria de los elementos del contenedor
T	inner_product	INIter1 primero1, INIter1 ultimo1, INIter2 primero2, T val_ini	Producto interno entre los contenedores mas val_ini

Contenedores asociativos: operaciones comunes (#include <set>, #include <map>)

A::insert(clave & x)	Inserta el elemento x en el contenedor
A::insert(A::iterator i, A::iterator f)	Inserta los elementos que están en el rango de los iteradores en el contenedor
A::erase(clave & x)	Borra todos los elementos que tengan la clave x
A::erase(A::iterator p)	Borra el elemento apuntado por p
A::count(clave & x)	Devuelve la cantidad de elementos que tienen la clave x
A::find(clave & x)	Devuelve un iterador al primer elemento que tenga la clave x
A::lower_bound(clave & x)	Devuelve un iterador al primer elemento que tenga la clave x
A::upper_bound(clave & x)	Devuelve un iterador al elemento siguiente al último elemento con clave x

map

M::operator[](clave & x)	Se utiliza para acceder al valor asociado a la clave x. Si no existe ningún elemento con esta clave, lo crea. EJ: M[4] = 1; *// inserta el par (4,1)*

Adaptadores: #include <stack>

empty()	Devuelve verdadero si la pila está vacía
size()	Retorna la cantidad de elementos de la estructura
push(T & x)	Inserta un elemento sobre el tope de la pila
pop()	Elimina el elemento que está en el tope de la pila
top()	Devuelve el elemento que está en el tope de la pila pero sin eliminarlo

Adaptadores: #include <queue>

empty()	Devuelve verdadero si la cola está vacía
size()	Retorna la cantidad de elementos de la estructura
push(T & x)	Inserta el elemento x al final de la cola
pop()	Elimina el elemento del frente de la cola

front()	Devuelve una referencia al frente de la cola
back()	Devuelve una referencia al final de la cola