STL (Standard - Template - Library)

Contenedores lineales

Pila: inserciones, consultas y borrados por sólo uno de los extremos (LIFO)

- Solo se puede acceder por tope (a_{n-1})

Funciones:
- size() -> num elementos
- empty() -> true si pila esta vacía
- top() -> elemento en tope
- pop() -> elimina por tope
- push() -> inserta por tope.

#include <stack>

Matriz:
- vector<vector<T>> m
- Podemos añadir class iterator para recorrer Matriz. (it_row, row_begin, row_end, col)

> Si utilizamos <T> hay que poner typename delante de declaración de iterator

Vector:
- size() -> size
- maxsize -> return maximum size
- resize (size_t n) ó resize (size_t n, const T& val) -> añade hasta n el valor val
- empty()
- operator []
- front() -> first element
- back() -> last element
- assign (iterator first, iterator last), assign(size_t n, valor) (n elementos con valor val), assign(list)
- push_back (valor) -> añade valor al final
- pop_back() -> elimina valor del final
- insert -> 1.(it, val) 2. (it, size_t n, val) 3. (Range) (pos, first, last) 4. (it, list)
- erase (it), erase(it_first, it_last)
- swap (vector) -> intercambia con vector
- clear()
- emplace() -> return iterator, emplace (it, args)

Escaneado con CamScanner

Listas: inserciones y borrados por cualquier posición {a1, ..., an}

- push_back(i) : añade al final de la lista
- reverse_iterator -> final a principio
- rbegin() -> último elemento válido de la lista
- rend() : elemento inicial lista
- size() : nº nodos lista
- reverse() : invierte una lista
- empty() : lista vacía
- front(): primer nodo
- back() : último nodo
- pop_front()
- pop_back()
- erase() <
  - iterator erase (iterator pos) : borra un elemento con su pos
  - iterator erase (iterator first, iterator last) : elimina desde first hasta end NO incluido.
- assign <
  - assign(it in, it fin) : asigna a lista valores en rango
  - assign (int, const T& v) : asigna a la lista el 2º parametro tantas veces como diga el int.
- insert <
  - insert (iterator it, int n, const T v) : inserta en la posición de it el objeto v tantas veces como diga n (it no varía)
  - insert (it, v) : inserta en it v => it apunta a v.
  - insert (it1, it2, it3) : inserta en it1 elementos desde it2 hasta it3
- swap (listdx) : intercambia
- clear() : lista vacía.
- splice (Convive de una lista a otra) <
  - splice (it, lista) : inserta a partir de it la lista, lista queda vacía.
  - splice (it_des, lista, it_source) : pone elemento apunta it_source de la lista que llama splice y lo pone donde apunta it_des.
  - splice (itdes, lista, itsource_ini, it_source_end)
- remove(val) : elimina de una lista elementos con valor val.
- remove_if(condición) : elimina elementos que cumplan condición
- unique() : elimina valores duplicados consecutivos (unique (función cond)) : estudia cond. a fun.
- sort(), • merge() mezcla dos listas ordenadas

Escaneado con CamScanner

Priority Queue: Colección de elementos ordenados por su prioridad o preferencia

#include <queue>

- size()
- empty()
- top() → elemento más prioritario
- pop() → elimina elem más prioritario
- push() → inserta un elemento en la pos dictada por su prioridad

Para definir prioridad tenemos que definir operator > sino, priondad se basa en qué número es mayor.

Deque (Doble Cola): se puede expandir y contraer por los dos extremos

Similares: a vectores pero con mejor eficiencia en los procesos de borrado e inserción

#include <deque>

Colas: siguen FIFO

- size()
- empty()
- front()
- push()
- back()
- pop()

Array: contenedores de tamaño fijo. Más el a vector en cuanto a almacenamiento - Vector estático.

- size()
- empty()
- operator []
- front()
- back()
- fill (val)
- swap (arrayX)

Escaneado con CamScanner

Forward_list: insertar y borrar en cualquier punto en tiempo cte.
Listas con celdas enlazadas. Forward_list mantiene un enlace al sgte elemento. Sólo se puede iterar hacia delante

#include <forward_list>

- empty()
- front()
- push_front() · pop_front()
- insert_after (it, val)

Contenedores asociativos

Set / Multiset: conjunto de elementos que se disponen de manera ordenada y en el que no se repiten elementos (set). Multiset admite elementos repetidos

- count (valor): num elementos iguales a valor (1 ó 0 en set >=0 en M.s)
- swap (set)
- find (elemento) → devuelve it con posición o encuentra, end().
- equal_range: pair<iterator, iterator> equal_range (const value_type & val) con range equivalent to val (beginning, end)
- lower_bound (val): devuelve iterator al primer elemento que coincide con val
- upper_bound (val): iterador al primer elemento mayor al val. Si no existe valor que no sea anterior → ∅ mayor más próximo
- Típicas (~~del~~, insert, empty, size ...)

Map / multimap: parejas de valores <clave, valor asociado>

map claves no rep, multimap claves rep.

mismas funciones que set/multiset + operator [].

Escaneado con CamScanner