

Operaciones para el tipo de dato string

<code>string s</code>	Constructor por defeto
<code>string s ("hola")</code>	Constructor con inicializador
<code>string s (aString)</code>	Constructor de copia
<code>s[i]</code>	Acceso al elemento i-ésimo del string
<code>s.substr(int pos,int len)</code>	Subcadena que comienza en pos y tiene longitud len
<code>s.c_str()</code>	Devuelve una cadena estilo C igual al string
<code>s.insert(int pos,string str)</code>	Insetar antes de pos el string str
<code>s.erase (int start, int len)</code>	Eliminar desde s[start] hasta s[start+len]
<code>s.replace(int start, int len,str)</code>	Sustituir desde s[start] hasta s[start+len] por str
<code>s.length()</code>	Longitud del string
<code>s.resize(int,char)</code>	Cambia el tamaño, rellenando con un valor
<code>s.empty()</code>	Cierto si el string es vacío
<code>s = s2</code>	Asignación de strings
<code>s += s2</code>	Concatenación de strings
<code>s + s2</code>	Nuevo string resultado de concatenar s y s2
<code>s ==s2 s != s2</code>	Igualdad y desigualdad de strings
<code>s < s2 s <= s2</code>	Comparaciones de strings (orden lexicográfico)
<code>s > s2 s >= s2</code>	Comparaciones de strings (orden lexicográfico)
<code>s.find(string str, int pos)</code>	Devuelve la posición en donde comienza la subcadena str desde s[pos].
<code>s.find_first_of(str,pos)</code>	Posición en donde se encuentra el primer carácter que pertenece a str desde s[pos].
<code>s.find_first_not_of(str,pos)</code>	Posición en donde se encuentra el primer carácter que no está en str desde s[pos].
<code>s.find_last_of(str,pos)</code>	Posición en donde se encuentra el último carácter que pertenece a str desde s[pos].
<code>s.find_las_not_of(str,pos)</code>	Posición en donde se encuentra el último carácter que no está en str desde s[pos].
<code>string::npos</code>	Valor entero retornado por find y sus variantes cuando no se encuentra la cadena buscada
Operaciones E/S	
<code>stream >> str</code>	Entrada de strings
<code>stream << str</code>	Salida de strings
<code>getline(stream,str,char)</code>	Añade a str todos los caracteres de una línea de la entrada estándar hasta encontrar el carácter char. Por defecto char es igual a '\n'.

Operaciones para el tipo de dato list

Constructores y asignación

<code>list<T> v</code>	Constructor por defecto
<code>list<T> l (aList);</code>	Constructor de copia
<code>l = aList</code>	Asignación

Acceso a elementos

<code>l.front()</code>	Primer valor de la colección
<code>l.back()</code>	Último valor de la colección

Inserción y borrado

<code>l.push_front (T)</code>	Añade un elemento al principio de la lista
<code>l.push_back (T)</code>	Añade un elemento al final de la lista
<code>l.insert (iterator, T)</code>	Inserta un nuevo elementos antes del iterador
<code>l.swap (list<T>)</code>	Intercambia valores con otra lista
<code>l.pop_front ()</code>	Borra el primer elemento de la lista
<code>l.pop_back ()</code>	Borra el último elemento de la lista
<code>l.remove(T)</code>	Eliminar todos los elementos iguales a uno dado
<code>l.remove_if(predicate)</code>	Eliminar todos los valores que cumplan una condición
<code>l.erase (iterator)</code>	Borra el elemento indicado por el iterador
<code>l.erase (iterator, iterator)</code>	Borra un rango de valores

Tamaño

<code>l.size ()</code>	Número de elementos en la lista
<code>l.empty ()</code>	Cierto si la lista está vacía

Iteradores

<code>list<T>::iterator itr</code>	Declara un nuevo iterador
<code>l.begin ()</code>	Iterador que referencia al primer elemento
<code>l.end ()</code>	Iterador que referencia al siguiente al último
<code>list<T>::reverse_iterator ritr</code>	Declara un nuevo reverse_iterator
<code>l.rbegin ()</code>	Reverse_iterator que referencia al último elemento
<code>l.rend ()</code>	Reverse_iterator que referencia al anterior al primero

Otros métodos

<code>l.reverse()</code>	Invierte la lista
<code>l.sort()</code>	Ordena los elementos de menor a mayor
<code>l.merge(list<T>)</code>	Mezcla con otra lista ordenada
<code>l.sort(comparision)</code>	Ordena los elementos según una función

Operaciones para los tipos de datos **vector** y **deque**

Constructores

<code>vector<T> v;</code>	Constructor por defecto
<code>vector<T> (int, T)</code>	Constructor con tamaño y valor inicial dados
<code>vector<T> v (aVector);</code>	Constructor de copia

Acceso a elementos

<code>v[i]</code>	Acceso por índice, también puede asignarse
<code>v.front()</code>	Primer valor de la colección
<code>v.back()</code>	Último valor de la colección

Inserción

<code>v.push_front (T)</code>	Añade un elemento al principio del vector (solo deque)
<code>v.push_back (T)</code>	Añade un elemento al final del vector
<code>v.insert (iterator, T)</code>	Inserta un nuevo elementos antes del iterador
<code>v.swap (vector<T>)</code>	Intercambia valores con otro vector

Borrado

<code>v.pop_front ()</code>	Borra el primer elemento del vector (solo deque)
<code>v.pop_back ()</code>	Borra el último elemento del vector
<code>v.erase (iterator)</code>	Borra el elemento indicado por el iterador
<code>v.erase (iterator, iterator)</code>	Borra un rango de valores

Tamaño

<code>v.capacity ()</code>	Número máximo de elementos del <i>buffer</i>
<code>v.size ()</code>	Número de elementos en el vector
<code>v.resize (unsigned, T)</code>	Cambia el tamaño, rellenando con un valor
<code>v.reserve (unsigned)</code>	Pone el tamaño del <i>buffer</i>
<code>v.empty ()</code>	Cierto si el vector está vacío

Iteradores

<code>vector<T>::iterator itr</code>	Declara un nuevo iterador
<code>v.begin ()</code>	Iterador que referencia al primer elemento
<code>v.end ()</code>	Iterador que referencia al siguiente al último
<code>vector<T>::reverse_iterator ritr</code>	Declara un nuevo reverse_iterator
<code>v.rbegin ()</code>	Reverse_iterator que referencia al último elemento
<code>v.rend ()</code>	Reverse_iterator que referencia al anterior al primero

Otros objetos y funciones útiles

```
back_inserter(contendor l); // genera un iterador especial para hacer que
                             algoritmos que reemplazan elementos inserten en lugar de reemplazar
istream_iterator<T>(istream &i); // iterador para leer elementos de tipo T desde
i
ostream_iterator<T>(ostream &o, const char *s); // iterador para escribir
                             elementos de tipo T en o separador por s
```

Algoritmos STL

f, l, pos denotan iteradores
 x, y denotan elementos

i, j, k denotan enteros
 p, q denotan funciones

Funcion	Descripción	Retorna
<code>advance(it,n)</code>	avanza el iterator n posiciones, modifica it	void
<code>next(it) / next(it,n)</code>	genera un nuevo iterator n (o 1) posiciones más adelante de it	iterator
<code>prev(it) / prev(it,n)</code>	genera un nuevo iterator n (o 1) posiciones más atrás de it	iterator
<code>distance(f,l)</code>	retorna cuantas posiciones hay desde f hasta l	size_t
<code>accumulate(f, l, i)</code>	suma todos los elementos entre f y l, iniciando el acumulador en i	typeof(i)
<code>count(f, l, x)</code>	cuenta y retorna las apariciones de x	size_t
<code>count_if(f, l, p)</code>	cuenta y retorna cuantos elementos satisfacen p	size_t
<code>equal(f1, l1, f2)</code>	determina si las secuencias son iguales	bool
<code>equal(f1, l1, f2, p)</code>	determina si las secuencias son iguales comparando con p	bool
<code>find(f, l, x)</code>	busca la primer ocurrencia del elemento x	iterator
<code>find_if(f, l, p)</code>	busca el primer elemento que cumpla con p	iterator
<code>generate(f, l, q)</code>	genera valores con la función q para los elementos entre f y l	void
<code>max_element(f, l)</code>	busca el mayor elemento en un rango dado	iterator
<code>max_element(f, l, p)</code>	busca el mayor elemento en un rango dado comparando con p	iterator
<code>min_element(f, l)</code>	busca el menor elemento en un rango dado	iterator
<code>min_element(f, l, p)</code>	busca el menor elemento en un rango dado comparando con p	iterator
<code>copy(f1, l1, f2)</code>	copia elementos desde f1...l1 a f2	ending position of output range
<code>fill(f1, l1, x)</code>	reemplaza todos en f1...l1 por x	
<code>fill_n(f1, n, x)</code>	reemplaza n elementos desde f1 por x	
<code>iter_swap(f1,f2)</code>	intercambia dos elementos	
<code>random_shuffle(f,l)</code>	reordena en forma aleatoria	
<code>remove(f, l, x)</code>	elimina todas las ocurrencias de x	iterator al comienzo de la parte que ya no se utiliza del contenedor
<code>remove_if(f, l, p)</code>	elimina todos los elementos que cumplen con p	iterator al comienzo de la parte que ya no se utiliza del contenedor
<code>replace(f, l, x, y)</code>	reemplaza x por y	
<code>replace_if(f, l, p, y)</code>	reemplaza los elementos que cumplan con p por y	
<code>reverse(f1, l1)</code>	invierte el orden	
<code>swap(x,y)</code>	intercambia dos elementos	
<code>swap_ranges(f1, l1, f2)</code>	intercambia los elementos de f1...l1 con los de f2...	iterator al final del segundo rango
<code>unique(f, l)</code>	remueve elementos repetidos (si está ordenado)	iterator al comienzo de la parte que ya no se utiliza del contenedor
<code>unique(f, l, p)</code>	idem al anterior pero comparando con p	iterator al comienzo de la parte que ya no se utiliza del contenedor
<code>binary_search(f, l, x)</code>	busca x entre f y l (solo si están ordenados, requier acc. aleat)	bool
<code>merge(f1, l1, f2, l2, f3)</code>	copia de f1...l1 y f2...l2 a f3... manteniendo el orden	iterator al final del rango f3...
<code>sort(f, l)</code>	ordena los elementos entre f y l	
<code>sort(f, l, p)</code>	ordena los elementos entre f y l comparando con p como <	

Char → metodos

isalnum	Comprobar si el carácter es alfanumérico	bool
isalpha	Check if character is alphabetic	bool
isblank	Comprobar si el carácter está en blanco ' \t', ' '.	bool
iscntrl	Comprobar si el carácter es un carácter de control '\n'	bool
isdigit	Comprobar si el carácter es un dígito decimal	bool
isgraph	Comprobar si el carácter tiene representación gráfica	bool
islower	Comprobar si el carácter es una letra minúscula	bool
isprint	Comprobar si el carácter es imprimible. Imprime carácter por carácter hasta que se encuentra con uno que no es imprimible. Ej: '\n'.	bool
ispunct	Comprobar si el carácter es un carácter de puntuación. Al igual que "isgraph" todos los que no sean "alfanuméricos".	bool
isspace	Comprobar si el carácter es un espacio en blanco. ' ', '\t', '\n', '\v', '\f'.	bool
isupper	Comprobar si el carácter es una letra mayúscula	bool
isxdigit	Comprobar si el carácter es un dígito hexadecimal.	bool

map

begin	Devuelve un Iterador al comienzo	Iterador
end	Devuelve un Iterador al final	Iterador
rbegin	Devuelve un Iterador al ultimo	Iterador
rend	Devuelve Iterador al comienzo	Iterador
cbegin	Devuelve un Iterador constante al comienzo	Iterador
cend	Devuelve el un Iterador constante al Final	Iterador
crbegin	Devuelve un Iterador constante al final	Iterador
crend	Devuelve un Iterador constante al comienzo	Iterador

Iterators

Capacidad

empty	Probar si el contenedor está vacío	size_type
size	Devuelve el tamaño del contenedor	
max_size	Devuelve el tamaño máximo del contenedor	

Acceso al Elemento

operator[]	Si k coincide con la clave de un elemento en el contenedor, la función devuelve una referencia a su valor asignado.	elemento
at	Devuelve una referencia al valor mapeado del elemento identificado con la clave k. <code>mymap.at("Ema") = 2022;</code>	&elemento

Modificadores

insert	Inserta un elemento. Si el elemento ya está en el contenedor y, de ser así, el elemento no se inserta, devolviendo un iterador a este elemento. Para insertar usar <code>operator[]</code>	Iterador al elemento insertado o al existente
erase	Elimina del contenedor del mapa un solo elemento o un rango de elementos ([primero, último]). Esto reduce efectivamente el tamaño del contenedor por el número de elementos eliminados, que se destruyen.	Int elementos borrados
swap	Intercambia el contenido del contenedor por el contenido de x, que es otro mapa del mismo tipo. Los tamaños pueden diferir.	none
clear		none

Operaciones:

find	Obtener iterador a elemento	iterador
count	Contar elementos con una clave específica	size_type
lower_bound	Devuelve un iterador que apunta al primer elemento del contenedor cuya clave no se k	iterador
upper_bound	Devuelve un iterador que apunta al primer elemento del contenedor cuya clave va despues de k	iterador
equal_range	La función devuelve un par, cuyo miembro par::primero es el límite inferior del rango (igual que límite_inferior), y par::segundo es el límite superior (igual que límite_superior).	pair::first pair::second