

THE CHAD

# Ordenamiento y Clasificación + Librería STD y sus estructuras de datos básicas

Por Ariel Parra.



**¿Qué es el ordenamiento en C++?**

El ordenamiento o en ingles sorting, son algoritmos que se utilizan para organizar los elementos de una colección en un orden específico.

los algoritmos mas comunes son los de la burbuja, seleccion, insercion, merge sort y quick sort.

# Burbuja Seleccion e Insercion

`<mark>Burbuja  $O(N^2)$ :</mark>` compara e intercambia elementos adyacentes si estan en el orden incorrecto.

`<mark>Seleccion  $O(N^2)$ :</mark>` busca el elemento mas pequeno y lo coloca en la posicion adecuada.

`<mark>Insercion  $O(N^2)$ :</mark>` construye una sublista ordenada y coloca los elementos restantes en su lugar.

# Merge sort y Quick sort

`<mark>Merge sort  $O(N\log(N))$ :</mark>` este divide la coleccion en mitades, ordena cada mitad y luego fusiona las mitades ordenadas.

`<mark>Quick sort  $O(N\log(N))$ :</mark>` selecciona un elemento llamado pivote y particiona la coleccion en dos partes ordenandolas por separado.

A collection of colorful geometric blocks and a sorting tray. The blocks include various shapes like octagons, squares, and triangles in colors such as blue, green, red, and orange. Some blocks are placed inside a white sorting tray with corresponding shaped holes. The brand name 'Melissa & Doug' is visible on the tray.

**¿Qué es la clasificación en C++?**

La clasificacion o busqueda son algoritmos que dividen un conjunto en categorias o clases en funcion de un criterio.

Los dos principales metodos de busqueda son los de busqueda lineal y la busqueda binaria.

# Busqueda lineal

La busqueda lineal recorre los elementos de la coleccion secuencialmente hasta encontrar el elemento buscado.

Si el elemento se encuentra se devuelve su posicion; de lo contrario se indica que no se encontro.

La complejidad de la busqueda lineal es  $O(N)$  donde  $N$  es el tamano de la coleccion.



# Busqueda binaria

Se aplica a colecciones ordenadas y divide repetidamente la busqueda a la mitad.

Se compara el elemento buscado con el elemento central de la coleccion y se descarta la mitad no deseada.

La complejidad temporal de la busqueda binaria es de  $O(\log(N))$ , donde  $n$  es el tamaño de la coleccion.

**¿Qué son las estructuras de datos?**

Son una forma organizacion de datos almacenados y elejidos dependiendo del uso y eficacia.

Los cuatro estructuras de datos que usaramos mas seguido en en la libreria std son los strings, vectores, sets y pairs.

**¿Qué es la librería STD de C++?**

Standard Library es una colección de clases y funciones, que crean la base del lenguaje C++, estas se basan en la Standard Template Library (STL).

Esta librería simplifica el uso y manipulación de varias estructuras de datos para poderlas usar con operadores booleanos y de asignación como si fuesen cualquier tipo de dato de C++.

librerías más usadas:

`<mark><iostream></mark>`: Funciones de entrada y salida estándar como el `std::cin` y `std::cout`.

`<mark><fstream></mark>` Permite el manejo de archivos.

`<mark><algorithm></mark>` Contiene los algoritmos más conocidos y eficientes, como `std::sort()` que usa una variación de quick sort.

`<mark><utility></mark>` utilidades y funciones generales especialmente el tipo `std::pair`.

`<mark><vector></mark>` Implementa arreglos dinámicos y funciones para manipularlos.

`<mark><string></mark>` Ofrece funcionalidades para manipular cadenas de caracteres, dándole funciones similares a las de los vectores.

`<mark><set></mark>` Implementa funciones para manipular sets.

# STD::string

son cadenas de caracteres de tamaño dinámico.

La complejidad de tiempo y espacio de sus funciones suele ser  $O(N)$  y  $O(1)$ .

<b>función</b>	<b>Descripción</b>
<code>string _nombre_;</code>	inicializa un string
<code>string _nombre_ = "_valor_";</code>	inicializa un string con un valor
<code>_nombre_.c_str();</code>	convierte el string std a string de c
<code>_nombre_.length();</code>	devuelve el tamaño del string
<code>_nombre_.empty();</code>	comprueba si está vacía
<code>_nombre_.clear();</code>	limpia el string
<code>_nombre_.insert(_posicion_,_string_);</code>	inserta un string en la posición dada
<code>_nombre_.append(_string_);</code>	agrega contenido al string
<code>_nombre_.compare(_string_);</code>	compara dos strings



# STD::vector

Gestiona automáticamente la memoria y almacena cualquier tipo de dato asignado.  
La complejidad de tiempo y espacio de sus funciones suele ser  $O(N)$  y  $O(1)$ .

función	Descripción
<code>vector &lt;_tipo_&gt; _nombre_;</code>	inicializa un vector
<code>vector &lt;_tipo_&gt; _nombre_(_tamaño_,<i>valorInicial</i>);</code>	inicializa un vector con un tamaño definido y valores iniciales
<code>_nombre_.size();</code>	devuelve el tamaño del vector
<code>_nombre_.empty();</code>	verifica si está vacío
<code>_nombre_.clear();</code>	limpia el vector
<code>_nombre_.pushback(_valor_);</code>	agrega un valor al final
<code>_nombre_.count(_valor_);</code>	cuenta cuántos elementos hay con ese valor

<b>función</b>	<b>Descripción</b>
<code>_nombre_.pop_back();</code>	elimina el ultimo elemento del vector
<code>_nombre_.insert(_posicion_,_valor_);</code>	inserta un valor en esa posicion
<code>_nombre_.erase(_posicion_);</code>	elimina un valor en la posicion dada
<code>_nombre_.swap(_otroVector_);</code>	intercambia el contenido con otro
<code>_nombre_.front();</code>	devuelve el primer elemento del vector
<code>_nombre_.back();</code>	devuelve el ultimo elemento del vector

## STD::set

Estos son vectores con elementos no repetidos y con un orden ascendente.  
La complejidad de tiempo y espacio de sus funciones suele ser  $O(\log(N))$ .

<b>función</b>	<b>Descripción</b>
<code>set &lt;_tipo_&gt; _nombre_;</code>	inicializa un set
<code>_nombre_.insert(_value_);</code>	inserta un valor al set
<code>_nombre_.erase(_valor_);</code>	elimina el valor dado si esta en el set
<code>_nombre_.empty()</code>	verifica si esta vacio
<code>_nombre_.clear();</code>	limpia el set
<code>_nombre_.count(_valor_);</code>	cuenta cuantos elementos hay (0 o 1)

# STD::pair

Un pair es una tipo de dato para variables con dos tipos de datos.

La complejidad de tiempo y espacio de sus funcinoes suele ser  $O(1)$ .

función	Descripción
<code>pair &lt;_tipo1_,_tipo2_&gt; _nombre_;</code>	inicializa un pair
<code>pair &lt;_tipo1_,_tipo2_&gt; _nombre_(_valor1_,_valor2_);</code>	inicializa un pair con valores iniciales
<code>_nombre_.first;</code>	devuelve el primer elemento
<code>_nombre_.second;</code>	devuelve el segundo elemento
<code>swap(_pair1_,_pair2_);</code>	intercambia sus contenidos (en si no es especifica de pair)

# Problemas de sets

236A: Boy or Girl

469A: I Wanna Be The Guy



# Referencias

[https://en.wikipedia.org/wiki/C%2B%2B\\_Standard\\_Library](https://en.wikipedia.org/wiki/C%2B%2B_Standard_Library)

[https://en.wikipedia.org/wiki/Standard\\_Template\\_Library](https://en.wikipedia.org/wiki/Standard_Template_Library)

<https://cplusplus.com/reference/string/string/>

<https://cplusplus.com/reference/vector/vector/>

<https://cplusplus.com/reference/set/set/>

<https://cplusplus.com/reference/utility/pair/>

[https://en.wikipedia.org/wiki/Data\\_structure](https://en.wikipedia.org/wiki/Data_structure)

<https://www.geeksforgeeks.org/sorting-algorithms/>