Maps y Hash en C++

Por Ariel Parra.

¿Qué es un map?


Rusia
Canadá
Estados Unidos de América
Kazajstán
China
Argelia
Libia
Brasil
Indonesia
Australia

Los mapas son contenedores asociativos que almacenan elementos de forma mapeada, tambien una implementación de una estructura de datos conocida como un árbol binario de búsqueda equilibrado (o árbol AVL) que almacena pares clave-valor dichos elementos se ordenan automáticamente por su clave (key) en orden ascendente. Cada elemento tiene un valor clave y un valor mapeado. Dos valores mapeados no pueden tener los mismos valores clave por lo que son unicas.

Map STD

El maps en STD estan incluidas en la libreria <map>, estos al estar equilibrados hacen la altura de los subárboles izquierdo y derecho de cualquier nodo difieran en un máximo de 1. Esto asegura un buen rendimiento en operaciones de búsqueda, inserción y eliminación.

La búsqueda en un std::map se realiza mediante la clave, lo que lo hace eficiente en términos de tiempo de búsqueda. Las operaciones de inserción y eliminación también son eficientes debido a la estructura equilibrada del árbol, en general tienen una complejidad de tiempo de O(log n), donde "n" es el número de elementos en el mapa. Dado que el árbol se mantiene equilibrado, el rendimiento es bastante constante para un gran número de elementos.

Operaciones STD

funcion	Descripcion
begin()	Devuelve un iterador al primer elemento del mapa.
end()	Devuelve un iterador al elemento teórico que sigue el último elemento en el mapa.
size()	Devuelve el número de elementos en el mapa.
max_size()	Devuelve el número máximo de elementos que el mapa puede contener.

funcion	Descripcion
empty()	Devuelve si el mapa está vacío.
par_insert(keyvalue, mapvalue)	Añade un nuevo elemento al mapa.
erase(itearador)	Elimina el elemento en la posición señalada por el iterador.
erase(const g)	Elimina el valor clave "g" del mapa.
clear()	Elimina todos los elementos del mapa.

Ejemplo

```
int main(){
map<string, int> mapa;

mapa["one"] = 1;
mapa["two"] = 2;
mapa["three"] = 3;

map <string, int>::iterator it = mapa.begin();

while (it != mapa.end()){
        std::cout << "Key: " << it->first << ", Value: " << it->second << std::endl;
        it++;
}

return 0;
}
```

¿Qué es un hash?

Los Hash A veces llamados diccionarios o tabla hash, es una técnica o proceso de asignación de claves, y valores en la tabla hash utilizando una función hash. Se hace para un acceso más rápido a elementos. La eficiencia del mapeo depende de la eficiencia de la función hash utilizada.

Las dos principales maneras de usar un hash es con maps y sets, en el caso de maps se llaman unordered maps y en el caso de sets se llaman unordered set.

Hash map

El unordered map es un contenedor asociado que almacena elementos formados por la combinación de un valor clave y un valor mapeado. El valor clave se utiliza para identificar el elemento y el valor mapeado es el contenido asociado con la clave. Tanto la clave como el valor pueden ser de cualquier tipo predefinido o definido por el usuario. En términos simples, un unordered_map es como una estructura de datos del tipo de diccionario que almacena elementos en sí mismo. Contiene pares sucesivos (key, valor), que permite la recuperación rápida de un elemento individual basado en su clave única.

Diferencias

Hash map	Map
Un Hash map key puede almacenarse en cualquier orden	El mapa es una secuencia ordenada de claves únicas
Hash map implementa una estructura de árbol desequilibrada debido a la cual no es posible mantener el orden entre los elementos	El mapa implementa una estructura de árboles equilibrada, por lo que es posible mantener el orden entre los elementos (por traversal de árboles específico)
La complejidad del tiempo de las operaciones son O(1) en promedio.	La complejidad del tiempo de las operaciones de mapa son O(log n)

Funciones	Descripción
at()	Esta función en C++ unordered_map devuelve la referencia al valor con el elemento como k clave
begin()	Devuelve un iterador apuntando al primer elemento en el contenedor en el contenedor unordered_map
end()	Devuelve un iterador señalando a la posición más allá del último elemento en el recipiente unordered_map
bucket()	Devuelve el número de cubo donde el elemento con el k clave se encuentra en el mapa
bucket_count()	se utiliza para contar el no total de cubos en el unordered_map. No se requiere ningún parámetro
bucket_size()	Devuelve el número de elementos en cada cubo de la unordered_map

Funciones	Descripción
count()	Contar el número de elementos presentes en unordered_map con una clave dada
equal_range()	Devuelve los límites de una gama que incluye todos los elementos en el contenedor con una clave que se compara igual a k
find()	Devuelve el iterador al elemento
empty()	Comprueba si el contenedor está vacío en el contenedor unordered_map
erase()	Borrar elementos en el contenedor en el contenedor unordered_map

Ejemplo

```
int main(){
  unordered_map<string, double> umap = {
  {"One", 1},
  {"Two", 2},
  {"Three", 3}
};

  // inserting values by using [] operator
  umap["PI"] = 3.14;
  umap["root2"] = 1.414;
  umap["root3"] = 1.732;
  umap["log10"] = 2.302;
  umap["loge"] = 1.0;
  umap.insert(make_pair("e", 2.718));
```

```
    string key = "PI";

    if (umap.find(key) == umap.end())
      cout << key << " not found\n\n";
    else
      cout << "Found " << key << "\n\n";

    key = "lambda";
    if (umap.find(key) == umap.end())
      cout << key << " not found\n";
    else
      cout << "Found " << key << endl;

    unordered_map<string, double>::iterator itr;
    cout << "\nAll Elements : \n";
    for (itr = umap.begin();
         itr != umap.end(); itr++)
    {
      cout << itr->first << "  " <<
              itr->second << endl;
    }
}
```

Output

```
Found PI

lambda not found

All Elements :
e  2.718
loge  1
log10  2.302
Two  2
One  1
Three  3
PI  3.14
root2  1.414
root3  1.732
```

Hash sets

Un hash set es un contenedor asociativo sin ordered implementado usando una tabla de hash donde las keys se introducen en índices de una tabla de hash para que la inserción sea siempre aleatorizada. Todas las operaciones son de tiempo constante O(1) en promedio y maximo de tiempo lineal O(n) en el peor caso que depende de la función hash usada.

Diferencias con Hash map

Unordered_map	Unordered_set
Unordered_map contiene elementos sólo en la forma de pares (valor clave).	Unordered_set no contiene necesariamente elementos en la forma de pares de valor clave, estos se utilizan principalmente para ver la presencia/absencia de un conjunto.
Operador ‘[]’ para extraer el valor correspondiente de una clave presente en el mapa.	La búsqueda de un elemento se realiza utilizando una función find(). Así que no hay necesidad de un operador[].

Diferencias con Set

Set	Unordered Set
A Set es una secuencia ordenada de claves únicas.	Un conjunto sin orden es un conjunto en el que se puede almacenar una clave en cualquier orden, tan sin orden.
El conjunto se implementa como una estructura de árboles equilibrada que permite mantener el orden entre los elementos (por traversal de árboles específico).	El conjunto unordered_set se implementa como tablas de hash ya que no tenemos que preocuparnos por ningún orden.
La complejidad de las operaciones de set es O(log n).	La complejidadde las operaciones de unordered_set es O(1).

Ejemplo

```
int main(){
        unordered_set<string> stringSet;

        stringSet.insert("code");
        stringSet.insert("in");
        stringSet.insert("c++");
        stringSet.insert("is");
        stringSet.insert("fast");

        string key = "slow";

        if (stringSet.find(key) == stringSet.end())
                cout << key << " not found" << endl << endl;
        else
                cout << "Found " << key << endl << endl;

        key = "c++";
        if (stringSet.find(key) == stringSet.end())
                cout << key << " not found\n";
        else
                cout << "Found " << key << endl;

        cout << "\nAll elements : ";
        unordered_set<string>::iterator itr;
        for (itr = stringSet.begin(); itr != stringSet.end();
                itr++)
                cout << (*itr) << endl;
}
```

output

```
slow not found

Found c++

All elements : is
fast
c++
in
code
```

Problemas

855A: Tom Riddle's Diary

443A: Anton and Letters

Referencias

https://youtu.be/_TYTIsTXk0w?si=v4ZBDI38MzI_Hn3I
https://www.youtube.com/watch?v=_TYTIsTXk0w
https://www.geeksforgeeks.org/map-associative-containers-the-c-standard-template-library-stl/
https://cppbyexample.com/map.html
https://www.geeksforgeeks.org/hashing-data-structure/
https://cppbyexample.com/hash_map.html
https://www.geeksforgeeks.org/unordered_set-in-cpp-stl/
https://www.geeksforgeeks.org/unordered_map-in-cpp-stl/