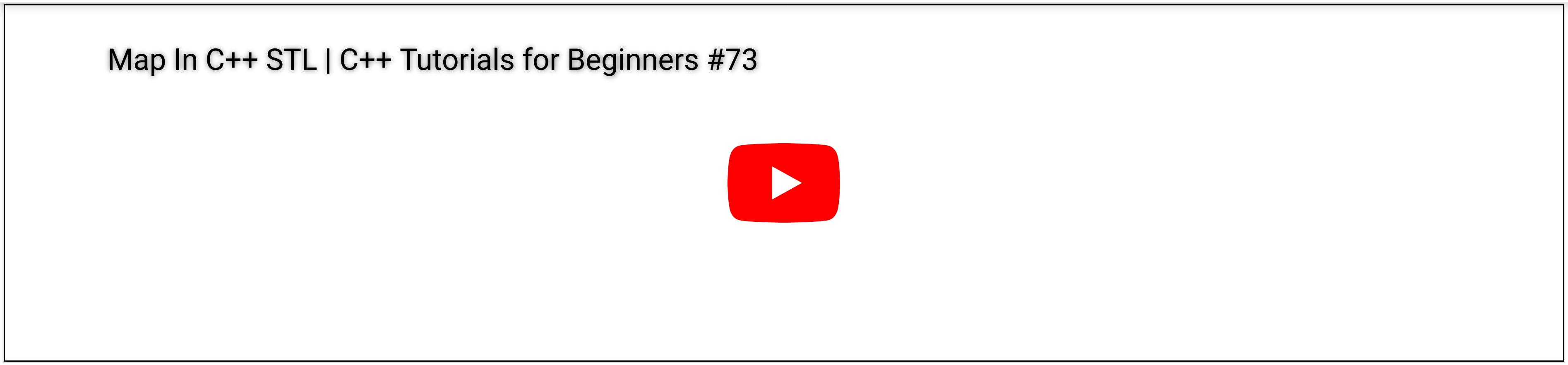
CodeWithHarry


Map In C++ STL | C++ Tutorials for Beginners #73

Overview Q&A Downloads Announcements

Map In C++ STL | C++ Tutorials for Beginners #73

So far, we have learned about vectors and lists in C++ STL, and today we will be learning about maps in C++ STL. It is important to clarify that whatever I have taught and whatever I will be teaching in the coming tutorials about STL isn't everything. And it is definitely not all. These are just the most important STL containers we will use. You have already seen how to explore more about STL from C++ - Containers.
We will now discuss maps, and because it is impractical to have every method on our fingers, I'd ask you all to also refer to the following website std::map - C++ Reference

A map in C++ STL is an associative container which stores key value pairs. To elaborate, a map stores a key of some data type and its corresponding values of some data type. For example: a teacher wants to store the marks of students which in future can be accessed by their names. Here, keys are the student names, and their marks are the corresponding values. Refer to the illustration below:

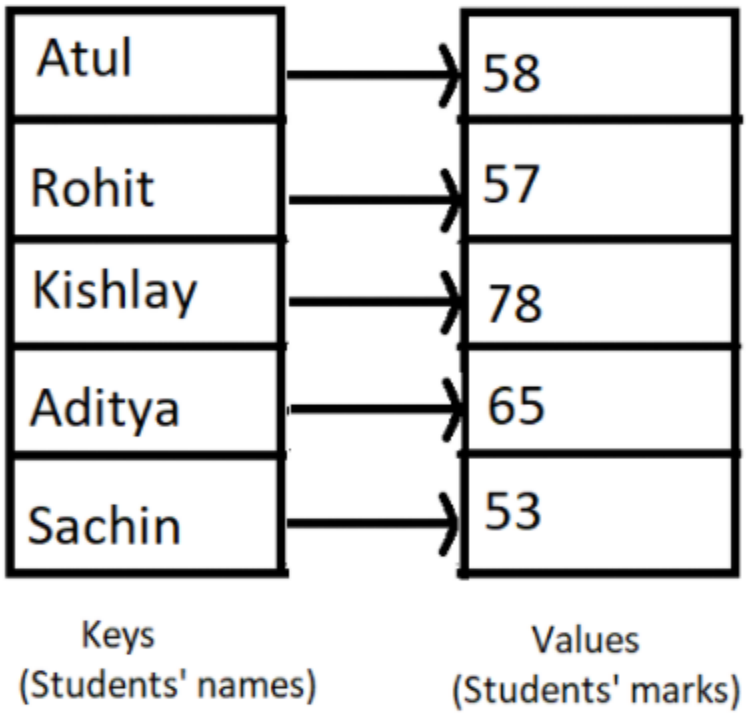


Figure 1: Illustrative diagram of a key value pairs

We can now shift to our editors and see how maps can be used in C++. Don't forget to include the header file <map>.

The syntax for declaring a map is:

```
map <data_type_of_key,   data_type_of_value>  variable_name;
```

Code Snippet 1: Syntax for declaring a map

And we can now write the program for storing the key value pairs of students' names and students' marks keeping in mind the illustration above. Refer to the snippet below.

Understanding code snippet 2:

1. Include the header file map and string(if using string methods).
2. Let's create a map in which the key is a string (names) and the values are integers (marks), and we'll call it marksMap.
3. And to assign some key a value, we use the index method. Here the index of a map element will be the students' name and the value will be the marks.
4. Make some 4-5 elements.
5. Identify the iterator of this map by using the scope resolution operator.
6. Loop through the map elements using two map methods; **begin()** to point at the beginning of the map, and **end()** to point next to the last element of the map.
7. While we loop through the map, we use the dereference operator * to fetch the element present where the pointer is pointing to. And since a map stores element in a key value pair, we can use its first and second method to access the keys and the values respectively. **.first** accesses the first value of a pair that is our map key here, and **.second** accesses the second value of the pair that is our map values here.
8. There is one thing to keep in mind: Maps always sort these pairs by the key elements. You can review the output of the following snippet to see how these pairs are sorted.

```
#include<iostream>
#include<map>
#include<string>

using namespace std;
```

```
int main(){

    // Map is an associative array
    map<string, int>  marksMap;
    marksMap["Atul"] = 58;
    marksMap["Rohit"] = 57;
    marksMap["Kishlay"] = 78;
    marksMap["Aditya"] = 65;
    marksMap["Sachin"] = 53;

    map<string,int> :: iterator iter;
    for (iter = marksMap.begin(); iter != marksMap.end(); iter++)
    {
        cout<<(*iter).first<<" "<<(*iter).second<<"\n";
    }


    return 0;
}
```

Code Snippet 2: A program to store names and marks using map

As you can see, the map pairs got sorted according to its key.

```
Aditya 65
Atul 58
Kishlay 78
```

```
Rohit 57
Sachin 53
PS D:\MyData\Business\code playground\C++ course>
```

Figure 2: Output of the above program

We have one more method to insert elements in a map. We can use **.insert()**

Syntax for using .insert is:

```
marksMap.insert({pair_1,pair_2.....pair_n});
```

Code Snippet 3: Syntax for inserting pairs in map

We will insert some elements into our map in snippet 2 by using the insert method.

```
 marksMap.insert( { {"Rohan", 89}, {"Akshat", 46} } );
```

Code Snippet 4: Program to insert two pairs in a map.

We can see the output to check if the above two pairs got inserted into the map or not.

```
Aditya 65
Akshat 46
Atul 58
Kishlay 78
Rohan 89
Rohit 57
Sachin 53
PS D:\MyData\Business\code playground\C++ course>
```

Figure 3: Output of the above program

So, yes, it worked. And the output was correct.

And I'd ask you all to explore more of these methods from the links I gave you above and start writing codes using them. This is the best way to learn. For example, you can use the **size()** method to get the size of the map container , **empty()** method to check if the map container is empty or not, and it returns a boolean. Therefore, this shouldn't be a big deal for you.

Thank you for being with me throughout the tutorial. I hope you enjoyed it. If you haven't checked out the whole playlist yet, move on to codewithharry.com or my YouTube channel to access it. You will surely enjoy them all. See you all in the next tutorial where we'll learn about Maps in C++ STL. Till then keep coding.

Previous Next

CodeWithHarry