

ThisPointer.Com

C, C++, Design Patterns, Datastructure, Algorithm, Multithreading Articles, Tutorials and Interview Questions

C++ 11 ▾

STL ▾

Multithreading


Datastructure ▾

Design Patterns

Home » C++ » std::map » Uncategorized » std::map Tutorial Part 1: Usage Detail with examples

std::map Tutorial Part 1: Usage Detail with examples

Varun January 31, 2015 C++, std::map, Uncategorized No Comment

 Like 4 people like this.

In this article we see how & why to use std::map in c++.

std::map Introduction

std::map is an associative container that store elements in key-value pair.

Benefits of using std::map :

- It stores only unique keys and that too in sorted order based on its assigned sorting criteria.
- As keys are in sorted order therefore searching element in map through key is very fast i.e. it takes logarithmic time.
- In std::map there will be only one value attached with the every key.
- std::map can be used as associative arrays.
- It might be implemented using balanced binary trees.

STL

STL

[std::vector](#)[std::set](#)[std::map](#)[STL Algorithm](#)[STL Interview Questions](#)

Advertisements

[Data Structure](#)[C++11 Threads](#)[Smart Pointer](#)[STL](#)

Binary Tree

Binary Search Tree

Linked List



Lets see an example,

```

1  #include <iostream>
2  #include <map>
3  #include <string>
4  #include <iterator>
5
6  int main()
7  {
8      std::map<std::string, int> mapOfWords;
9      // Inserting data in std::map
10     mapOfWords.insert(std::make_pair("earth", 1));
11     mapOfWords.insert(std::make_pair("moon", 2));
12     mapOfWords["sun"] = 3;
13     // Will replace the value of already added key i.e
14     mapOfWords["earth"] = 4;
15     // Iterate through all elements in std::map
16     std::map<std::string, int>::iterator it = mapOfWor
17     while(it != mapOfWords.end())
18     {
19         std::cout<<it->first<<" :: "<<it->second<<std:
20         it++;
21     }
22     // Check if insertion is successful or not
23     if(mapOfWords.insert(std::make_pair("earth", 1)).s
24     {
25         std::cout<<"Element with key 'earth' not inser
26     }
27     // Searching element in std::map by key.
28     if(mapOfWords.find("sun") != mapOfWords.end())
29         std::cout<<"word 'sun' found"<<std::endl;
30     if(mapOfWords.find("mars") == mapOfWords.end())
31         std::cout<<"word 'mars' not found"<<std::endl;
32     return 0;
33 }
```

Output:

earth :: 4

moon :: 2

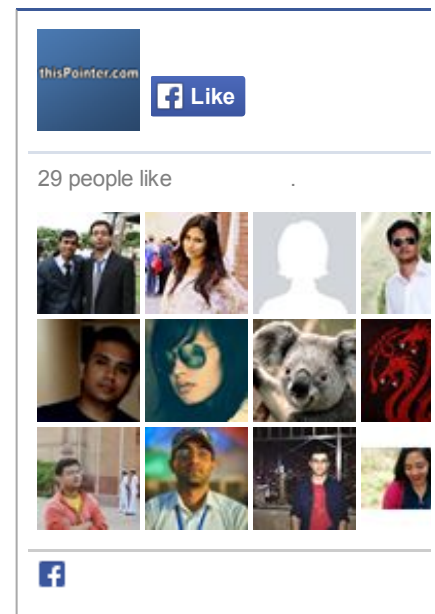
sun :: 3

Element with key 'earth' not inserted because already existed

word 'sun' found

word 'mars' not found

Find Us On Facebook



Advertisements

AdChoices

[▶ C++ Interview Questions](#)

[▶ C++ Example](#)

[▶ C++ Tutorial](#)

Advertisements



Creating std::map objects

Creating a std::map of words i.e.

Key = Word (std::string)

Value = Word's frequency count (int)

```
1 | std::map<std::string, int> mapOfWords;
```

As no external sorting criteria for key(std::string) is specified in above std::map, therefore it will use default key sorting criteria i.e operator < and all elements will be arranged inside std::map in alphabetical sorted order of keys.

Inserting data in std::map :

Inserting data using insert member function,

```
1 | mapOfWords.insert(std::make_pair("earth", 1));  
2 | mapOfWords.insert(std::make_pair("moon", 2));
```

We can also insert data in std::map using operator [] i.e.

```
1 | mapOfWords["sun"] = 3;
```

Different between operator [] and insert function:

If specified key already existed in map then operator [] will silently change its value where as insert will not replace already added key instead it returns the information i.e. if element is added or not. e.g.

Live Traffic

```
1 | mapOfWords["earth"] = 4; // Will replace the value of a
```

Where as for insert member function,

```
1 | mapOfWords.insert(std::make_pair("earth", 1)).second
```

will return false.

Iterating through all std::map elements:

```

1  std::map<std::string, int>::iterator it = mapOfWords.begin();
2  while(it != mapOfWords.end())
3  {
4      std::cout<<it->first<<" :: "<<it->second<<std::endl;
5      it++;
6  }

```

Each entry in `std::map<std::string, int>` is `std::pair<std::string, int>` therefore through iterator, key can be accessed by `it->first` and value by `it->second`.

Searching element in std::map by key

`find` member function of `std::map` can be used to search element in `std::map` by key. If specified key is not present then it returns the `std::map::end` else an iterator to the searched element.

```

1  iterator find (const key_type& k);
2
3  //e.g.
4
5  if(mapOfWords.find("sun") != mapOfWords.end())
6      std::cout<<"word 'sun' found"<<std::endl;
7  if(mapOfWords.find("mars") == mapOfWords.end())
8      std::cout<<"word 'mars' not found"<<std::endl;

```

Searching element in std::map by Value

To search element in `std::map` by value we need to iterate through all of the elements and check for the passed value and return i.e.


```


1  #include <iostream>
2  #include <map>
3  #include <string>
4  #include <iterator>
5
6  std::map<std::string, int>::iterator searchByValue(std::map<std::string, int> &mapOfWords, int val)
7  {
8      // Iterate through all elements in std::map and search for the value
9      std::map<std::string, int>::iterator it = mapOfWords.begin();
10     while(it != mapOfWords.end())
11     {
12         if(it->second == val)
13             return it;
14         it++;
15     }
16 }
17 int main()
18 {
19     std::map<std::string, int> mapOfWords;


```


Live Traffic Feed


[Watch Live](#) | [Get FEEDJIT](#)


 Milpitas, California, United States arrived 0 secs ago.


 Bjärred, Skåne, Sweden arrived 6 mins ago.


 Stone Mountain, Georgia, United States arrived 16 mins ago.


 Swidnik, Lublin Voivodeship, Poland arrived 18 mins ago.


 Brazil arrived 29 mins ago.

 Brazil arrived 1 hour 24 mins ago.

 Netherlands arrived 2 hours 33 mins ago.

 Catanzaro, Calabria, Italy arrived 3 hours 38 mins ago.

 Tokyo, Tōkyō, Japan arrived 4 hours 4 mins ago.

 Kumar, Himachal Pradesh, India arrived 4 hours 28 mins ago.

```

20 // Inserting data in std::map
21 mapOfWords.insert(std::make_pair("earth", 1));
22 mapOfWords.insert(std::make_pair("moon", 2));
23 mapOfWords["sun"] = 3;
24
25 std::map<std::string, int>::iterator it = serachBy
26 if(it != mapOfWords.end())
27     std::cout<<it->first<<" :: "<<it->second<<std:
28
29 return 0;
30 }

```

Output:

sun :: 3

Deleting data from std::map

std::map's erase member function is used to delete the element in std::map i.e.

```

1 void erase (iterator position);
2 size_type erase (const key_type& k);
3 void erase (iterator first, iterator last);

```

Code example,

```

1 #include <iostream>
2 #include <map>
3 #include <string>
4 #include <iterator>
5 int main()
6 {
7     std::map<std::string, int> mapOfWords;
8     mapOfWords.insert(std::make_pair("earth", 1));
9     mapOfWords.insert(std::make_pair("moon", 2));
10    mapOfWords["sun"] = 3;
11
12    // Erasing By iterator
13    std::map<std::string, int>::iterator it = mapOfWor
14    mapOfWords.erase(it);
15
16    // Erasing By Key
17    mapOfWords.erase("earth");
18
19    return 0;
20 }

```

Related Posts

Differences between a pointer and reference

Set vs Map : How to choose a right associative container ?

Iterating over a range of User Defined objects and calling member function

std::for_each Tutorial : Usage Details with Examples

Importance of Constructors while using User Defined Objects with

No Comments Yet

 [Subscribe to comments feed](#)

Leave a Reply

Your email address will not be published. Required fields are marked *

Name ***Email *****Website****Comment**

You may use these HTML tags and attributes: <abbr title=""> <acronym title=""> <blockquote cite=""> <cite> <code> <del datetime=""> <i> <q cite=""> <s> <strike>

Post Comment

« [Create a Binary Search Tree from an array](#)

[std::map Tutorial -Part 2: std::map and External Sorting Criteria / Comparator](#) »

[Terms and Conditions](#)
[Policy](#)