C++ : Map Tutorial Part 1: Usage Detail with examples

Varun January 31, 2015 C++, std::map 1 Comment

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.

Lets see an example,

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

int main()
{
    std::map<std::string, int> mapOfWords;
    // Inserting data in std::map
    mapOfWords.insert(std::make_pair("earth", 1));
    mapOfWords.insert(std::make_pair("moon", 2));
    mapOfWords["sun"] = 3;
    // Will replace the value of already added key i.e. earth
    mapOfWords["earth"] = 4;
    // Iterate through all elements in std::map
    std::map<std::string, int>::iterator it = mapOfWords.begin();
    while(it != mapOfWords.end())
    {
        std::cout<<it->first<<" :: "<<it->second<<std::endl;
        it++;
    }
    // Check if insertion is successful or not
    if(mapOfWords.insert(std::make_pair("earth", 1)).second == false)
    {
        std::cout<<"Element with key 'earth' not inserted because already existed"<<std::endl;
    }
    // Searching element in std::map by key.
    if(mapOfWords.find("sun") != mapOfWords.end())
        std::cout<<"word 'sun' found"<<std::endl;
    if(mapOfWords.find("mars") == mapOfWords.end())
        std::cout<<"word 'mars' not found"<<std::endl;
    return 0;
}
```

Output:

earth :: 4
moon :: 2
sun :: 3
Element with key 'earth' not inserted because already existed

Vector List Deque Set Map MultiMap STL Algorithms

1.) What's std::vector and why to use it?
2.) Different ways to initialize a vector
3.) How does std::vector works internally
4.) User Defined Objects & std::vector
5.) How to use vector efficiently in C++?
6.) std::vector and Iterator Invalidation
7.) Remove repeated elements from a vector
8.) Fill a vector with random numbers
9.) Hidden cost of std::vector
10.) Adding elements in Vector

word 'sun' found
word 'mars' not found

Creating std::map objects

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

Key = Word (std::string)
Value = Word's frequency count (int)

```
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,

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

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

```
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.

```
mapOfWords["earth"] = 4; // Will replace the value of already added key.
```

Where as for insert member function,

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

will return false.

Iterating through all std::map elements:

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

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.

```
iterator find (const key_type& k);

//e.g.

if(mapOfWords.find("sun") != mapOfWords.end())
std::cout<<"word 'sun' found"<<std::endl;
if(mapOfWords.find("mars") == mapOfWords.end())
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.

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

std::map<std::string, int>::iterator serachByValue(std::map<std::string, int> & mapOfWords, int val
{
    // Iterate through all elements in std::map and search for the passed element
    std::map<std::string, int>::iterator it = mapOfWords.begin();
    while(it != mapOfWords.end())
    {
        if(it->second == val)
        return it;
        it++;
    }
}
int main()
{
    std::map<std::string, int> mapOfWords;
    // Inserting data in std::map
    mapOfWords.insert(std::make_pair("earth", 1));
    mapOfWords.insert(std::make_pair("moon", 2));
    mapOfWords["sun"] = 3;

    std::map<std::string, int>::iterator it = serachByValue(mapOfWords, 3);
    if(it != mapOfWords.end())
        std::cout<<it->first<<" :: "<<it->second<<std::endl;

return 0;
}
```

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.

```
void erase (iterator position);
size_type erase (const key_type& k);
void erase (iterator first, iterator last);
```

Code example,

```
#include <iostream>
#include <map>
#include <string>
#include <iterator>
int main()
{
    std::map<std::string, int> mapOfWords;
    mapOfWords.insert(std::make_pair("earth", 1));
    mapOfWords.insert(std::make_pair("moon", 2));
    mapOfWords["sun"] = 3;

    // Erasing By iterator
    std::map<std::string, int>::iterator it = mapOfWords.find("moon");
    mapOfWords.erase(it);

    // Erasing By Key
    mapOfWords.erase("earth");

    return 0;
}
```

Other Map related articles are,

1.) std::map Usage Detail with examples

2.) std::map and Comparator

3.) std::map & User defined class objects as keys

4.) Set vs Map

5.) How to Iterate over a map in C++

6.) Map Insert Example

Search

Search ... Search

7.) Iterate a map in reverse order

8.) Check if a key exists in a Map

9.) Search by value in a Map

10.) Erase by Key | Iterators

11.) C++ Map : Operator []

12.) Erase by Value or callback

Click Here to Subscribe for more Articles / Tutorials like this.

C++, std::map, STL

1 Comment Already

Warren - May 3rd, 2016 at 7:20 am

Thanks for the clear demonstration.

Reply

Leave a Reply

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

Name *

Email *

Website

Post Comment

« Create a Binary Search Tree from an array

C++ : Map Tutorial -Part 2: Map and External Sorting Criteria / Comparator »

C++11 – Utilities

std::bind
auto specifier
Variadic Templates

Pointers

Pointer vs Refrence
Allocating 2D Array Dynamically

STL – Vector

1.) What's std::vector and why to use it?
2.) Different ways to initialize a vector

Java – HashSet

Subscribe

STL Multimap

MultiMap Example and Tutorial

multimap::equals_range – Tutorial

STL Algorithms

std::sort Tutorial & Example

std::unique Tutorial & Example

1.) Using std::find & std::find_if with User Defined Classes

2.) Iterating over a range of User Defined objects and calling member function using std::for_each

Terms and Conditions

Terms and Conditions Policy