

thisPointer.com

[C++11 Tutorials](#)[STL](#)[C++ Tutorials](#)[Multithreading](#)[Python](#)[Boost Library ▾](#)[Design Patterns](#)[java](#)[GDB](#)[Datastructure](#)[Subscribe](#)[Binary Search Tree](#)[Binary Tree](#)[Linked List](#)

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,

```
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. earth
14    mapOfWords["earth"] = 4;
15    // Iterate through all elements in std::map
16    std::map<std::string, int>::iterator it = mapOfWords.begin();
17    while(it != mapOfWords.end())
18    {
19        std::cout<<it->first<<" :: "<<it->second<<std::endl;
20        it++;
21    }
22    // Check if insertion is successful or not
23    if(mapOfWords.insert(std::make_pair("earth", 1)).second == false)
24    {
25        std::cout<<"Element with key 'earth' not inserted because already existed"<<std::endl;
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

[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

Advertisements

Advertisements

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)

```
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.

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

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

UTS POSTGRAD EXPO

16 APRIL

REGISTER NOW

22095

Subscribe For latest Tutorials

★ indicates required

Email Address ★

Subscribe

Advertisements

Advertisements

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 serachByValue(std::map<std::string, int> & mapOfWords, int val
7  {
8      // Iterate through all elements in std::map and search for the passed element
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;
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 = serachByValue(mapOfWords, 3);
26     if(it != mapOfWords.end())
27         std::cout<<it->first<<" :: "<<it->second<<std::endl;
28
29     return 0;
30 }

```

Search

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 = mapOfWords.find("moon");
14     mapOfWords.erase(it);
15
16     // Erasing By Key
17     mapOfWords.erase("earth");
18
19     return 0;
20 }

```

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](#)

- [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.](#)

Like 454

Share

 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](#) »

Like us on Facebook

Like

Share

454 people like this. Be tl

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

What is Hashing and Hash Table?
Create and add elements in a HashSet
Iterate over a HashSet
Search for an element in HashSet
Merge two HashSets
Initializing HashSet from an Array
Convert a HashSet into an Array
Merge an Array in a HashSet

Java Interview Questions

Method Overriding Tutorial
Overriding with Different Return Type
Calling Base class's overridden method
Preventing Method Overriding
Need of preventing Method Overriding

Subscribe

Subscribe For latest Tutorials

* indicates required

Email Address *

Subscribe

Design Patterns

Behavioral Design Patterns
Observer Design Pattern
State Design Pattern
Strategy Design Pattern
Structural Design Patterns
Composite Design Pattern
Flyweight Design Pattern
Creational Design Patterns
Factory Method Design Pattern

C++11 – Unordered Set

- 1.) unordered_set Basic Example
- 2.) Initializing an unordered_set
- 3.) Inserting elements in an unordered_set
- 4.) Searching an element in unordered_set
- 5.) unordered_set – Custom Hasher & Comparator
- 6.) Unordered_set & User defined classes

C++11 – UnorderedMap

Basic Usage Detail and Example
Initializing an unordered_map
Searching in unordered_map
Insert elements in unordered_map
Erasing an element
Erase elements while iterating
std::map vs std::unordered_map

C++11 Smart Pointers

shared_ptr<> Tutorial and Examples
shared_ptr and Custom Deletor
shared_ptr vs raw pointer
Create shared_ptr objects carefully
weak_ptr Tutorial | shared_ptr and Cyclic References
unique_ptr<> Tutorial and Examples

C++11 Multithreading

Part 1: Three Ways to Create Threads
Part 2: Joining and Detaching Threads
Part 3: Passing Arguments to Threads
Part 4 : Sharing Data & Race Conditions
Part 5 : Fixing Race Conditions using mutex
Part 6 : Need of Event Handling
Part 7: Condition Variables
Part 8: std::future and std::promise
Part 9: std::async Tutorial & Example
Part 10: std::packaged_task<> Tutorial

C++11 Rvalue References

lvalue vs rvalue
Is rvalue immutable in C++?
What is rvalue reference in C++11
Move Constructor

Callbacks in C++

Function Pointers
Function Objects & Functors

C++ Strings

Find and Replace all occurrences of a string
Find all occurrences of a sub string
Case Insensitive string::find
Convert First Letter of each word to Upper Case
Converting a String to Upper & Lower Case
Trim strings in C++
C++ : How to split a string using String and character as Delimiter?
startsWith() Implementation
endsWith() Implementation
Remove Sub Strings from String

C++ Memory Management

Memory Leaks
new and delete operator
delete vs []delete
Out Of Memory Errors
Overload new & delete
Restrict Dynamic Deletion
Placement new operator
Delete 'this' pointer

Polymorphism

Virtual Functions
vTable and vPointer

- 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

STL – Deque

- 1.) What is std::deque and how deque works internally.
- 2.) deque vs vector : What to choose ?

STL – List

- 1.) std::list Internals & Usage Details
- 2.) List vs Vector
- 3.) Different ways to Initialize a list
- 4.) Erase elements using iterators
- 5.) Remove elements while Iterating
- 6.) Remove elements based on External Criterion
- 7.) Get element by index in List
- 8.) Searching an element in std::list
- 9.) Different Ways to iterate over a List
- 10.) Sorting a List & custom Comparator

STL – Set

- 1.) C++ Set basic example and Tutorial
- 2.) Using std::set with user defined classes
- 3.) std::set and external Sorting criteria | Comparator
- 4.) Access Element by index in Set
- 5.) How to insert elements in Set
- 6.) How to iterate over a Set
- 7.) Removing an element from Set
- 8.) Erase elements while Iterating & Generic erase_if()

STL – Map

- 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
- 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
- 13.) copy all Values from a Map to vector

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

