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

Varun January 31, 2015 C++, std::map 6 Comments

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>
```


Advertisements

```
#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 == f
    {
        std::cout<<"Element with key 'earth' not inserted becaus
    }
    // 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
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.

Entity LS15 Safety LE...
Polygon XQUARONE...
Advertisements
Get 28% off electricity a gas usage r plus $100 credit
NSW only. Conditions apply.
EnergyAustralia
LIGHT THE WAY
Sign up now

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 ad
```

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

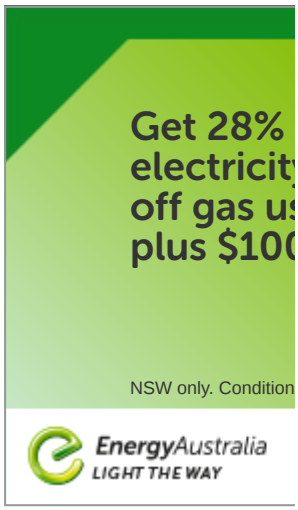
Subscribe For latest Tutorials

* indicates required

Email Address *

Subscribe

Advertisements



Advertisements

Search

Search … Search

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:
{
    // Iterate through all elements in std::map and search for t
    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(mapO
    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("m
    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

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 466 Share

C++, std::map, STL

6 Comments Already

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

Thanks for the clear demonstration.

Reply

Varun - April 20th, 2018 at 8:32 am

Thanks For appreciating.

Reply

Pat Patterson - April 18th, 2018 at 4:22 am

How do I do something straightforward like get the value associated with at looked-up key and assign it to a variable?

Reply

Varun - April 20th, 2018 at 8:32 am

Method 1 :
If you are sure that key exists in map then directly access using [] operator i.e.

value = mapOfWords[key];

Method 2 :

If you are not sure that key exists in map then,

```
auto it = mapOfWords.find(key);
if(it != mapOfWords.end())
{
value = it->second;
}
```

Reply

Mario - June 22nd, 2018 at 1:59 am

What if the key value type doesn't have a default operator< ?

Reply

Varun - June 24th, 2018 at 12:25 pm

Then you can pass custom comparators. Checkout following articles for complete examples,
std::map and Comparators
std::map and User define objects

Reply

Leave a Reply

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

Name *

Email *

Website

Save my name, email, and website in this browser for the next time I comment.

Post Comment

uses Akismet to reduce spam. Learn how your comment data is processed.

« Create a Binary Search Tree from an array

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

Python : List

Check if an item exists in List

Check if a list contains all the elements of other list

Create a List and initialize with values

How to Iterate over a List

Insert an element at specific index in List

Sort a list of tuples by 2nd Item

Sort a list of strings

Add an element in list | append() vs extend()

Check if all elements in a List are same

Merge / Join two or more lists

Remove Duplicates from a List

Convert a list to string

Remove element from a list by value or Index

Remove multiple elements from list

Python : Dictionary

Creating Dictionaries in Python

C++11 – Utilities

std::bind

auto specifier

Variadic Templates

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

Advertisement

High Resolution Aerial Maps - N Date Images of Aust

Save Site Visits. Frequently Updated Ima For Businesses. Free Trial

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

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

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

Iterating over Dictionaries in Python

Check if a key exists in Dictionary

Get list of all the keys in Dictionary

Get list of all the Values in a Dictionary

Remove multiple keys in Dictionary while Iterating

Remove a key from Dictionary

Add key/value pairs in Dictionary

Find keys by value in Dictionary

Sort a Dictionary by key or Value

Copy a dictionary | Shallow vs Deep Copy

Python Strings

Access characters in string by index in Python

Iterate over the characters in string

How to Replace characters in a string ?

Java – Hashmap

What is Hashing and Hash Table?

Associating Multiple values with same Key

Remove elements while Iterating

Update the value of an existing key

Get all keys by a value in HashMap

Java – HashSet

What is Hashing and Hash Table?

Create and add elements in a HashSet

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

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 Manangement

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

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

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

lvalue vs rvalue

Is rvalue immutable in C++?

What is rvalue reference in C++11

Move Contsructor

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

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

Copyright ©2018. thispointer.com