thispointer.com

C++ Map Insert Example

Varun September 21, 2016 insert, std::map, STL 1 Comment

In this article we will discuss how to insert a key value pair in std::map.

Map internally store elements in a pair of key value i.e.

```
std::pair<key Type, Value Type>
```

So, to add an element in map we can use one of its member function insert() i.e.

```
pair<iterator,bool> insert (const value_type& element);
```

It accepts an object of key value pair and returns an pair of map iterator and bool.

In the returned pair i.e.

```
pair<iterator,bool>
```

bool represents the result of insertion and iterator represents the position of newly added element in map.

If insertion in map is successful then,
bool —> true
Iterator —-> Points to Position of newly added element.

If insertion in map failed then,

bool —> false

Iterator —-> Pointing to the passed pair

Let's see an example,

Suppose we have a map of string & int as key value pair i.e.

```
std::map<std::string, int> mapOfWordCount;
```

To insert an element in it we will create a pair of string & int i.e.

```
std::pair<std::string, int>("first", 1)
```

insert() function will return a pair of map iterator & bool i.e.

```
std::pair<std::map<std::string, int>::iterator, bool > result;
result = mapOfWordCount.insert(std::pair<std::string, int>("firs
```

Now, to check if insertion we will first check the bool in result pair i.e.

```
if(result.second == false)
{
   std::cout<<"Failed to add . duplicate key :: "<<result.first->
}
else
{
   std::cout<<"Successful in Adding , key :: "<<result.first->fi
}
```

Complete example is as follows,

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

void testResult(
        std::pair<std::map<std::string, int>::iterator, bool> &
    // Check if Insertion was successful
    if (result.second == false) {
        // Insertion Failed
        std::cout << "Failed to add . duplicate key :: " << resu
                << std::endl;
    } else {
        // Insertion was successful
        std::cout << "Successful in Adding , key :: " << result.
                << std::endl;
    }
}

int main() {
```

```
22
23      // Map of string and int
24      std::map<std::string, int> mapOfWordCount;
25
26      // Pair of Map Iterator and bool
27      std::pair<std::map<std::string, int>::iterator, bool> result
28
29      // Insert Element in map
30      result = mapOfWordCount.insert(std::pair<std::string, int>("
31      // Test its result
32      testResult(result);
33
34      // Insert Element in map
35      result = mapOfWordCount.insert(std::pair<std::string, int>("
36      // Test its result
37      testResult(result);
38
39      // Insert Element in map
40      result = mapOfWordCount.insert(std::pair<std::string, int>("
41      // Test its result
42      testResult(result);
43
44      // Try to add duplicate element
45      result = mapOfWordCount.insert(std::pair<std::string, int>("
46      // Test its result
47      testResult(result);
48
49      // Create a map iterator and point to beginning of map
50      std::map<std::string, int>::iterator it = mapOfWordCount.beg
51
52      std::cout << "*****************************" << std::endl;
53      // Iterate over a map using std::for_each and Lambda functio
54      std::for_each(mapOfWordCount.begin(), mapOfWordCount.end(),
55              [](std::pair<std::string, int> element) {
56                  // Accessing KEY from element
57                  std::string word = element.first;
58                  // Accessing VALUE from element.
59                  int count = element.second;
60                  std::cout<<word<<" :: "<<count<<std::endl;
61              });
62
63      return 0;
64  }
```

Output:

```
1  Successful in Adding , key :: first
2  Successful in Adding , key :: second
3  Successful in Adding , key :: third
4  Failed to add . duplicate key :: third
5  *****************************
6  first :: 1
7  second :: 2
8  third :: 3
```

To compile the above example in linux use following command,

g++ -std=c++11 example.cpp

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

Related Posts:

- How to Iterate over a map in C++
- How to Insert elements in an unordered_set in C++11
- Different ways to insert elements in an unordered_map
- C++ : Different ways to insert elements in Set
- How to Access Element by index in a Set | C++
- How to Sort a Map by Value in C++
- C++ : How to find an element in vector and get its index ?
- Finding all values for a key in multimap using...
- C++ : How to find duplicates in a vector ?
- multimap Example and Tutorial in C++
- How to iterate a map in reverse order – C++
- Unordered_map Usage Tutorial and Example
- How to iterate over an unordered_map in C++11
- C++11 'auto' Tutorial and Examples
- C++ List – Find | Contains : How to search an...
- How to copy all Values from a Map to a Vector in C++
- How to search by value in a Map | C++
- How to Erase / Remove an element from an unordered_map
- Python : How to add / append key value pairs in...
- Different Ways to initialize an unordered_map
- C++ map : Erase element by key or Iterator or Range
- C++11 – std::all_of() – Algorithm...
- C++11 : std::any_of() Examples and Tutorial
- How to find an element in unordered_map
- C++ : How to insert element in vector at specific...

1 Comment Already

Aniket Bhardwaj - December 3rd, 2017 at 7:07 pm

Great Article. As part of code optimization can be skip these additional steps as below:
// Pair of Map Iterator and bool
std::pair<std::map::iterator, bool> result;

// Insert Element in map
result = mapOfWordCount.insert(std::pair(“first”, 1));
// Test its result
testResult(result);

and replace it with one line code:
testResult(mapOfWordCount.insert(std::pair(“first”, 1)));

Please let me know if i am missing something.

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

« How to Iterate over a map in C++

How to iterate a map in reverse order – C++ »

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

Iterating over Dictionaries in Python

Check if a key exists in Dictionary

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

Insert elements in unordered_map

Erasing an element

Erase elements while iterating

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

Callbacks in C++

Function Pointers

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

1.) std::list Internals & Usage Details

2.) List vs Vector

3.) Different ways to Initialize a list

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

Iterate over a HashSet

Search for an element in HashSet

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

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

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

Merge two HashSets

Initializing HashSet from an Array

Convert a HashSet into an Array

Merge an Array in a HashSet

Move Contsructor

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

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

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