unordered\_map in C++ STL

unordered\_map is an associated container that stores elements formed by combination of key value and a mapped value. The key value is used to uniquely identify the element and mapped value is the content associated with the key. Both key and value can be of any type predefined or user-defined.

Internally unordered\_map is implemented using [Hash Table](https://www.geeksforgeeks.org/hashing-set-1-introduction/), the key provided to map are hashed into indices of hash table that is why performance of data structure depends on hash function a lot but on an average the cost of **search, insert and delete** from hash table is O(1).

// C++ program to demonstrate functionality of unordered\_map

#include <iostream>

#include <unordered\_map>

using namespace std;

int main()

{

// Declaring umap to be of <string, int> type

// key will be of string type and mapped value will

// be of int type

unordered\_map<string, int> umap;

// inserting values by using [] operator

umap["GeeksforGeeks"] = 10;

umap["Practice"] = 20;

umap["Contribute"] = 30;

// Traversing an unordered map

for (auto x : umap)

cout << x.first << " " << x.second << endl;

}

**Output:**

Contribute 30

GeeksforGeeks 10

Practice 20

**unordered\_map vs [unordered\_set](https://www.geeksforgeeks.org/unorderd_set-stl-uses/) :**  
In [**unordered\_set**](https://www.geeksforgeeks.org/unorderd_set-stl-uses/), have only key, no value, these are mainly used to see presence/absence in a set. For example, consider the problem of counting frequencies of individual words. We can’t use [**unordered\_set**](https://www.geeksforgeeks.org/unorderd_set-stl-uses/) (or set) as we can’t store counts.

**Methods on unordered\_map:**  
A lot of function are available which work on **unordered\_map**.

most useful of them are operator **=**, operator **[ ]**, **empty** and **size** **for capacity**, **begin** and **end** **for iterator**, **find** and **count** **for lookup**, **insert** and **erase** **for modification**.  
The C++11 library also provides function to see internally used bucket count, bucket size and also used hash function and various hash policies but they are less useful in real application.  
We can iterate over all elements of **unordered\_map** using Iterator.

**Initialization, indexing and iteration** is shown in below :

// C++ program to demonstrate functionality of unordered\_map

#include <iostream>

#include <unordered\_map>

using namespace std;

int main()

{

// Declaring umap to be of <string, double> type

// key will be of string type and mapped value will

// be of double type

unordered\_map<string, double> umap;

// inserting values by using [] operator

umap["PI"] = 3.14;

umap["root2"] = 1.414;

umap["root3"] = 1.732;

umap["log10"] = 2.302;

// inserting value by insert function

**umap.insert( make\_pair(** "e", 2.718 **)** **) ;**

string key = "PI";

// If key not found in map iterator to end is returned

if (**umap** . **find(** key **)** == **umap**.**end()** )

cout << key << " not found\n\n";

// If key found then iterator to that key is returned

else

cout << "Found " << key << "\n\n";

key = "lambda";

if (**umap** . **find(** key **)** == **umap**. **end()** )

cout << key << " not found\n";

else

cout << "Found " << key << endl;

// iterating over all value of umap

**unordered\_map<string, double>::** **iterator** itr;

cout << "\nAll Elements : \n";

for (itr = **umap.** **begin();** itr != **umap .end();** itr++)

{

// itr works as a pointer to pair<string, double>

// type itr->first stores the key part and

// itr->second stroes the value part

cout << **itr->first** << " " << **itr->second** << endl;

}

}

**Output:**

Found PI

lambda not found

All Elements :

e 2.718

log10 2.302

root3 1.732

PI 3.14

root2 1.414

**A practical problem based on unordered\_map** – given a string of words, find frequencies of individual words.

Input : str = "geeks for geeks geeks quiz practice qa for";

Output : Frequencies of individual words are

(practice, 1)

(for, 2)

(qa, 1)

(quiz, 1)

(geeks, 3)

// C++ program to find freq of every word using unordered\_map

#include <bits/stdc++.h>

using namespace std;

void printFrequencies(const string &str)

{

// declaring map of <string, int> type, each word

// is mapped to its frequency

**unordered\_map < string, int >** **wordFreq** ;

// breaking input into word using string stream

**stringstream** ss( str ) ; // Used for breaking words

string word; // To store individual words

while ( ss >> word)

{ wordFreq[ word ]++; }

unordered\_map< string, int> :: **iterator** p;

for (p = wordFreq.begin(); p != wordFreq.end(); p++)

cout << "(" << p->first << ", " << p->second << ")\n";

}

int main()

{

string str = "geeks for geeks geeks quiz "

"practice qa for";

printFrequencies( str );

return 0;

}

**Methods of unordered\_map :**

* [at()](https://www.geeksforgeeks.org/std-unordered_map-c/): This function in C++ unordered\_map returns the reference to the value with the element as key k.
* [begin()](https://www.geeksforgeeks.org/unordered_map-begin-in-c/): Returns an iterator pointing to the first element in the container in the unordered\_map container
* [end()](https://www.geeksforgeeks.org/unordered_map-end-function-in-c-stl/): Returns an iterator pointing to the position of the last element in the container in the unordered\_map container
* [bucket():](https://www.geeksforgeeks.org/unordered_map-bucket-in-c-stl/) Returns the bucket number where the element with the key k is located in the map.
* [bucket\_count:](https://www.geeksforgeeks.org/stdbucket_count-stdbucket_size-unordered_map-cpp/) bucket\_count is used to count the total no. of buckets in the unordered\_map. No parameter is required to pass into this function.
* [bucket\_size:](https://www.geeksforgeeks.org/stdbucket_count-stdbucket_size-unordered_map-cpp/) Returns number of elements in each bucket of the unordered\_map.
* [count()](https://www.geeksforgeeks.org/unordered_map-count-in-c/): Count the number of elements present in an unordered\_map with a given key.
* [equal\_range](https://www.geeksforgeeks.org/unordered_map-equal_range-in-c/): Return the bounds of a range that includes all the elements in the container with a key that compares equal to k.