Determine if map contains a value for a key?

What is the best way to determine if a STL map contains a value for a given key?

```
#include <map>

using namespace std;

struct Bar
{
    int i;
};

int main()
{
    map<int, Bar> m;
    Bar b = {0};
    Bar b1 = {1};

    m[0] = b;
    m[1] = b1;

    //Bar b2 = m[2];
    map<int, Bar>::iterator iter = m.find(2);
    Bar b3 = iter->second;

}
```

Examining this in a debugger, it looks like `iter` is just garbage data.

If I uncomment out this line:

```
Bar b2 = m[2]
```

The debugger shows that `b2` is `{i = 0}` . (I'm guessing it means that using an undefined index will return a struct with all empty/uninitialized values?)

Neither of these methods is so great. What I'd really like is an interface like this:

```
bool getValue(int key, Bar& out)
{
    if (map contains value for key)
    {
        out = map[key];
        return true;
    }
    return false;
}
```

Does something along these lines exist?

c++ stl map

edited Aug 25 '14 at 3:48 — jww — 34.9k 21 109 221

asked Jun 28 '10 at 22:10 — Nick Heiner — 35.8k 113 364 615

1 possible duplicate of How to find if a given key exists in a C++ std::map – OrangeDog Jul 24 '14 at 12:33

7 Answers

> Does something along these lines exist?

No. With the stl map class, you use `::find()` to search the map, and compare the returned iterator to `std::map::end()`

so

```
map<int,Bar>::iterator it = m.find('2');
Bar b3;
if(it != m.end())
{
   //element found;
   b3 = it->second;
}
```

Obviously you can write your own `getValue()` routine if you want (also in C++, there is no reason to use `out`), but I would suspect that once you get the hang of using `std::map::find()` you won't want to waste your time.

Also your code is slightly wrong:

`m.find('2');` will search the map for a keyvalue that is `'2'`. IIRC the C++ compiler will implicitly convert '2' to an int, which results in the numeric value for the ASCII code for '2' which is not what you want.

Since your keytype in this example is `int` you want to search like this: **`m.find(2);`**

edited Oct 20 '12 at 21:24
user283145

answered Jun 28 '10 at 22:14

Alan
26.6k 12 80 113

3 Yeah, good catch with the `'2'` v. `2` error. – Nick Heiner Jun 28 '10 at 22:21

1 This answers sounds much better: stackoverflow.com/a/11765524/496223 – dynamic May 23 '13 at 8:22

4 How so? `find` indicates intent far better than `count` does. More over, `count` doesn't return the item. If you read the OP's question, he's wants to check for the existance, *and* return the element. `find` does that. `count` does not. – Alan May 23 '13 at 16:49

if (m.count(key)) b3 = m[key]; //or whatever – pconnell Jul 16 '13 at 19:25

8 I've always been curious as to what kind of weed were smoking the people who designed the whole stl API. – Trap Oct 8 '14 at 15:40

|

As long as the map is not a multimap, one of the most elegant ways would be to use the count method

```
if (m.count(key))
    // key exists
```

The count would be 1 if the element is indeed present in the map.

edited Mar 5 '14 at 23:29

vPraetor
113 2 11

answered Aug 1 '12 at 18:43

pconnell
1,864 1 7 4

3 Surprised it's not the accepted answer.. – vines Oct 5 '12 at 0:44

7 Won't this check *all* the keys even if it has found one already? That can get expensive fast... – mmdanziger Oct 24 '12 at 20:42

16 It will only count more than one key if used on a multimap. – Andrew Prock Mar 13 '13 at 23:38

9 @mmdanziger No, it won't be expensive: cplusplus.com/reference/map/map/count Count is logarithmic in size. – jgyou Nov 2 '14 at 17:46

11 The key exists, and then what? At that point you'd usually want to get the value for it, paying for another search (e.g. using `operator[]`). `find` gives you .NET's `TryGetValue` semantics, which is almost always what you (and specifically the OP) want. – Ohad Schneider Nov 6 '14 at 15:58

|

It already exists with find only not in that exact syntax.

```
if (m.find(2) == m.end() )
{
    // key 2 doesn't exist
}
```

If you want to access the value if it exists, you can do:

```
map<int, Bar>::iterator iter = m.find(2);
if (iter != m.end() )
{
    // key 2 exists, do something with iter->second (the value)
}
```

With C++0x and auto, the syntax is simpler:

```
auto iter = m.find(2);
if (iter != m.end() )
{
    // key 2 exists, do something with iter->second (the value)
}
```

I recommend you get used to it rather than trying to come up with a new mechanism to simplify it. You might be able to cut down a little bit of code, but consider the cost of doing that. Now you've introduced a new function that people familiar with C++ won't be able to recognize.

If you want to implement this anyway in spite of these warnings, then:

```
template <class Key, class Value, class Comparator, class Alloc>
bool getValue(const std::map<Key, Value, Comparator, Alloc>& my_map, int key,
Value& out)
{
    typename std::map<Key, Value, Comparator, Alloc>::const_iterator it =
my_map.find(key);
    if (it != my_map.end() )
    {
        out = it->second;
        return true;
    }
    return false;
}
```

answered Jun 28 '10 at 22:16

stinky472
5,404 17 24

`amap.find` returns `amap::end` when it does not find what you're looking for -- you're supposed to check for that.

answered Jun 28 '10 at 22:13

Alex Martelli
476k 87 865 1146

Check the return value of `find` against `end` .

```
map<int, Bar>::iterator iter = m.find('2');
if ( map.end() != iter ) {
  // contains
  ...
}
```

answered Jun 28 '10 at 22:13

JaredPar
450k 85 919 1225

You can create your getValue function with the following code:

```
bool getValue(const std::map<int, Bar>& input, int key, Bar& out)
{
   std::map<int, Bar>::iterator foundIter = input.find(key);
   if (foundIter != input.end())
   {
      out = foundIter->second;
```

```
        return true;
    }
    return false;
}
```

edited Apr 8 at 22:50

netjeff
5,481 4 16 26

answered Jun 28 '10 at 22:21

Kip Streithorst
319 2 7

I believe line 6 should be `out = foundIter->second` – Dithermaster Jun 29 '14 at 16:50

I fixed Kip's answer to correctly show `out = foundIter->second` rather than `out = *foundIter` – netjeff Apr 8 at 22:52

If you want to determine whether a key is there in map or not, you can use the find() or count() member function of map. The find function which is used here in example returns the iterator to element or map::end otherwise. In case of count the count returns 1 if found, else it returns zero(or otherwise).

```
if(phone.count(key))
{ //key found
}
else
{//key not found
}

for(int i=0;i<v.size();i++){
    phoneMap::iterator itr=phone.find(v[i]);//I have used a vector in this example
to check through map you cal receive a value using at() e.g: map.at(key);
    if(itr!=phone.end())
        cout<<v[i]<<"="<<itr->second<<endl;
    else
        cout<<"Not found"<<endl;
}
```

answered Jul 30 at 8:05

Prashant Shubham
11 7