C++
Information
Tutorials
Reference
Articles
Forum

Forum
Beginners
Windows Programming
UNIX/Linux Programming
General C++ Programming
Lounge
Jobs

How to define assign member function in map ?

zesan (26) Jan 18, 2018 at 1:11pm

I got the following task with limited time for submission, so far I have done the solution but not sure is there anything wrong what must be corrected before I submit, please help me out. Sorry for detail task description.

Task Description

interval_map<K,V> is a data structure that efficiently associates intervals of keys of type K with values of type V. Your task is to implement the assign member function of this data structure, which is outlined below.

interval_map<K, V> is implemented on top of std::map. In case you are not entirely sure which functions std::map provides, what they do and which guarantees they provide, we provide an excerpt of the C++1x draft standard here: "Here other file I didn't paste"

Each key-value-pair (k,v) in the m_map member means that the value v is associated to the interval from k (including) to the next key (excluding) in m_map.

Example: the std::map (0,'A'), (3,'B'), (5,'A') represents the mapping

0 -> 'A'
1 -> 'A'
2 -> 'A'
3 -> 'B'
4 -> 'B'
5 -> 'A'
6 -> 'A'
7 -> 'A'
... all the way to numeric_limits<key>::max()

The representation in m_map must be canonical, that is, consecutive map entries must not have the same value: ..., (0,'A'), (3,'A'), ... is not allowed. Initially, the whole range of K is associated with a given initial value, passed to the constructor.

Key type K

besides being copyable and assignable, is less-than comparable via operator<
is bounded below, with the lowest value being std::numeric_limits<K>::lowest()
does not implement any other operations, in particular no equality comparison or arithmetic operators

Value type V

besides being copyable and assignable, is equality-comparable via operator==
does not implement any other operations

You are given the following source code:

```
1
2  #include <assert.h>
3  #include <map>
4  #include <limits>
5  #include <iostream> // I added this was not given am I doing right ?
6
7  using namespace std;
8
9  template<class K, class V>
10 class interval_map {
11     friend void IntervalMapTest();
12
13 private:
14     std::map<K,V> m_map;
15
16 public:
17     // constructor associates whole range of K with val by inserting (K_min, val)
18     // into the map
19     interval_map( V const& val) {
20         m_map.insert(m_map.begin(),std::make_pair(std::numeric_limits<K>::lowest(),val));
21     }
22
23     // Assign value val to interval [keyBegin, keyEnd).
24     // Overwrite previous values in this interval.
25     // Do not change values outside this interval.
26     // Conforming to the C++ Standard Library conventions, the interval
27     // includes keyBegin, but excludes keyEnd.
28     // If !( keyBegin < keyEnd ), this designates an empty interval,
29     // and assign must do nothing.
30
31
32
33     void assign( K const& keyBegin, K const& keyEnd, V const& val ) {
34
35  // My code strats
```

```
36
37 if(!(keyBegin < keyEnd))
38     return;
39
40  //int  it;
41
42  //std::map<std::string, int>::iterator it =  m_map.insert();
43
44         typename std::map<K,V>::iterator it =m_map.begin();
45
46
47   for(int j=keyBegin; j<keyEnd; j++)
48   {
49     it = m_map.find(j);
50
51     if(it==m_map.end())
52     {
53       m_map.insert(pair<K,V>(j,val));
54     }
55     else
56     {
57       m_map.erase(it);
58       m_map.insert(pair<K,V>(j, val));
59     }
60   }
61 //end of my code
62
63  }
64
65
66
67     // look-up of the value associated with key
68     V const& operator[]( K const& key ) const {
69         return ( --m_map.upper_bound(key) )->second;
70     }
71
72
73 // my code again
74 void IntervalMapTest()
75      {
76
77           std::map<char,unsigned int> m_map;
78   char c;
79
80   m_map ['A']=0;
81   m_map ['A']=1;
82   m_map ['A']=2;
83
84   for (c='A'; c<'B'; c++)
85   {
86     std::cout << c;
87     if (m_map.count(c)>0)
88       std::cout << " is an element of mymap.\n";
89     else
90       std::cout << " is not an element of mymap.\n";
91   }
92
93      }
94 //me code ends
95
96
97
98 };
99
100 // Many solutions we receive are incorrect. Consider using a randomized test
101 // to discover the cases that your implementation does not handle correctly.
102 // We recommend to implement a function IntervalMapTest() here that tests the
103 // functionality of the interval_map, for example using a map of unsigned int
104 // intervals to char.
105
106 int main() {
107
108 // given IntervalMapTest();
109
110 // What I did starts
111
112 interval_map <char, unsigned int> test1 ('K');
113
114     // instantiation with float and char type
115
116  interval_map <unsigned int, char> test2 (5);
117
118     test1.IntervalMapTest();
119
120      test2.IntervalMapTest();
121
122 // What I did ends
123
124 }
125
```

Last edited on Jan 20, 2018 at 10:06am

dhayden (3192) Jan 18, 2018 at 6:09pm

Line 41 assumes that K and V are std::string and int respectively.

Your loop from lines 45-58 may work but it won't leave the map in canonical order. Also,
if you ad an interval from, say 1 to 3 billion, it's going to take a while to insert all 3 billion entries. It seems to me that the point of this exercise is to insert just one entry.

Here's my initial thought on how to approach assign().

If necessary, adjust the map so the interval ending at KeyBegin is correct.
If necessary, adjust the map so the interval beginning at KeyEnd is correct.
Remove existing intervals in the range [KeyBegin, KeyEnd)
Insert the new range for [KeyBegin, KeyEnd)
Finally, canonicalize it: if the range before and/or after [KeyBegin, KeyEnd) has the same value as the new range, then combine them.

As I type this, it occurs to me that for the last part, you might actually start by checking the current value for KeyBegin is equal to the value being inserted. If so, change KeyBegin to be the beginning of the existing range. Do the same with KeyEnd. Now you're guaranteed that the result will be canonical.

Note that the point here is to start by figuring out how to do this on paper. Don't start writing code until you have figured out the algorithm.

zesan (26) Jan 19, 2018 at 5:07pm

Thanks a lot for the reply, actually it would be very kind if anyone can help me out by providing corrected code on existing one since running out of time to figure out by myself, I am sorry for making such a request.

ne555 (9270) Jan 20, 2018 at 11:03am

> not sure is there anything wrong what must be corrected before I submit
It seems that you quite misunderstood the assignment.
«We recommend to implement a function IntervalMapTest() here that tests the functionality of the interval_map» however your `IntervalMapTest()' member (¿?) function makes no mention to the `interval_map' class or any of its members, and such, you had missed flagrant errors like `m_map.insert()` (glad that you fixed it)
I would expect something like

```
interval_map<int, std::string> m("foo");
m.assign(2,5) = "hello";
m.assign(7,9) = "world";
for K in range[0, 10:
   print(m[K])
//more intervals and printing
```

(they actually suggest a randomized test, but start here. also, think some corner cases)

«Key type K does not implement any other operations, in particular no equality comparison or arithmetic operators» and yet you do `for(int j=keyBegin; j<keyEnd; j++)`
as dhayden said, you are not supposed to insert every number in the interval
and `int' is not good enough to hold whatever a key may be.

Look at the example you were given
To represent the intervals

```
[-inf, 0) -> default
[0, 3) -> 'A'
[3, 5) -> 'B'
[5, +inf] -> 'A'
```

the map only needs to be `std::map (0,'A'), (3,'B'), (5,'A')`
just three elements.

Let's say that you want to insert the interval `[1,2) -> 'x'`
first, realise that it goes inside the interval [0, 3), so that will be broken in

```
[0, 1) -> 'A'
[1, 2) -> 'x'
[2, 3) -> 'A'
```

to do that, you insert into the map

```
m[1] = 'x'; //note, m[keyBegin] = val
m[2] = 'A'; //note, m[keyEnd] = ¿how do you obtain the `A' value?
```

suppose that you want to insert `[4, 7) -> 'o'`
that's between [3, 5) and [5, +inf] and would end

```
[3, 4) -> 'B'
[4, 7) -> 'o'
[7, +inf) -> 'A'
```

so you need to do

```
m[4] = 'o' //again, m[keyBegin] = val
m[7] = 'A' //again, m[keyEnd] = ¿how do you obtain the `A' value?
m.erase(5) //delete everything that's between [keyBegin, keyEnd). look at lower_bound() and upper_bound()
```

> help me out by providing corrected code on existing one since running out of
> time to figure out by myself
if you can't do it in the time limit, then fail the assignment
don't worry too much about it.

Last edited on Jan 20, 2018 at 11:05am

Topic archived. No new replies allowed.