

## Re: use remove\_if in map

---

*Source:* <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.stl/2005-07/msg00036.html>

---

- *From:* "Tom Widmer [VC++ MVP]" <tom\_usenet@xxxxxxxxxxxx>
  - *Date:* Tue, 12 Jul 2005 11:15:42 +0100
- 

wangbo wrote:

Hif- Can I use remove\_if in map,I write some codes like this,but it can't compile,why:

```
class RemoveInvalid
{
public:
    RemoveInvalid(std::set<int> validset)
    {
        m_validset = validset;
    };
    std::set<int> m_validset;
    bool operator ()(std::pair<int,int> con)
    {
        if(m_validset.find(con.first) != m_validset.end())
            return false;
        return true;
    };
};
...
std::map<int,int> m_con_defnum;
...
m_con_defnum.erase(std::remove_if(m_con_defnum.begin(),m_con_defnum.end(),Re
moveInvalid(validcon)),m_con_defnum.end());
```

You cannot use mutating algorithms on `std::map` or `std::set`, essentially because the operation `*i = val` is illegal for iterators of those containers. In any case, `remove_if` is only a good algorithm for containers that aren't node based, like `std::vector` and `std::deque`. For `std::list`, `std::list::remove_if` is more efficient. If you had a set, you could use `set_difference` to do what you want. With `std::map`, you have two options. The first is to iterate over the valid set and copy the valid map elements into another map (possibly via a vector, to easily allow the use of the map's iterator range constructor). This is  $O(s * \log m)$  where  $s$  is the size of the valid set, and  $m$  that of the map. Option 2 is to iterate over the map and check each element for validity. This is  $O(m * \log(s))$  (and it basically what you have been trying to do). In plain code:

## Re: use remove\_if in map

```
for(std::map<int,int>::iterator i = m_con_defnum.begin(),
    end = m_con_defnum.end();
    i != end;)
{
    if (validset.find(i->first) == validset.end())
    {
        m_con_defnum.erase(i++);
    }
    else
    {
        ++i;
    }
}
```

Note the use of post-increment on `i` - this ensures that `i` is incremented before it is invalidated by the erase operation.

Tom

.