

Re: How does STL map deal with scoping and memory persistence

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.vc/2005-07/msg00431.html>

- *From:* "Carl Daniel [VC++ MVP]" <cpdaniel_remove_this_and_nospam@xxxxxxxxxxxxxxxxxxx>
 - *Date:* Mon, 18 Jul 2005 23:09:05 -0700
-

Tommo wrote:

```
> I am storing some key value pairs in a map as follows:
>
> <code>
> bool StaticDataCache::putMarket(uint8_t key,item_t* value)
> {
> typedef pair<uint8_t,item_t*> entry;
> typedef map<uint8_t,item_t*>::iterator iter;
>
> entry newEntry(key,value);
> cout << "About to insert with key:" << key << endl;
> pair<iter,bool> insertSuccess = marketsMap.insert(newEntry);
>
> return true;
> }
> </code>
>
> Now suprisingly to me this actually works when i was expecting it not
> to. As you may notice the pair , newEntry, is local to this function,
> which means, as i understand it, that once this function exits, the
> memory used by newEntry is reclaimed and hence the pair inserted into
> the map would be invalidated. Can anyone tell me why this is working.
> Does the STL pair handle a 'new' behind the scenes??
```

All STL containers are value based. That means (among other things), then whenever you add something to an STL container, the item is copied into memory allocated and managed by the container. Your local copy then has no link to the copy in the container and can be destroyed immediately.

-cd

Re: How does STL map deal with scoping and memory persistence

- *Follow-Ups:*

- ◆ *Re: How does STL map deal with scoping and memory persistence*

- ◇ *From:* Tommo

- *References:*

- ◆ *How does STL map deal with scoping and memory persistence*

- ◇ *From:* Tommo

- Prev by Date: *Drawing an Image over Another*

- Next by Date: *Re: How does STL map deal with scoping and memory persistence*

- Previous by thread: *How does STL map deal with scoping and memory persistence*

- Next by thread: *Re: How does STL map deal with scoping and memory persistence*

- Index(es):

- ◆ *Date*

- ◆ *Thread*