Subject: std::map not treated properly
Posted by Bertram Kopf on Wed, 21 Jan 2009 13:51:25 GMT
View Forum Message <> Reply to Message

Dear all,
I took a brief look over the code and I noticed that the usage of the std::map(s) are partly not treated in a correct way. In particular maps which contains pointers as a key are used without any compare operators.

Here are just two examples:

in recotasks/TrackFitStatTask.cxx: "std::map<CbmMCTrack*,int> mctruthmap;"

in emc/EmcData/PndEmcDigi.cxx: "std::map<PndEmcTwoCoordIndex*, PndEmcXtal*> ..."

In these cases the map uses the actual pointer values for the sorting. This results in a randomly ordered map which is in addition very error prone. It's also not possible to decide that two key pointers containing exactly the same thing are equal.
Therefore I would propose to define a base class/structure as a template which defines such "less" comparisons. In addition each key object has to have a "less" operator.

Best regards,
Bertram.

Subject: Re: std::map not treated properly
Posted by Mohammad Al-Turany on Fri, 23 Jan 2009 13:22:46 GMT
View Forum Message <> Reply to Message

Hallo Bertram,

Theoretically and in principle you are right that when std::map is used with pointers as keys the less operator should be implemented! and usually this is what you read in any STL book with the example of const char pointer and two pointer pointing to the same set of character and then the MAP can not decide it self which one is bigger! but here this is completely meaningless and will never introduce errors unless some body come to the idea to compare two tracks (e.g if (CbmMCTrack *t1 < CbmMCTrack *t2)) what ever this comparison means!

and even if we try to implement this less operator what does it mean here how can a track be less or larger than another?

I think the important thing here is the IsEqual, but this we have as we inherits from TObject so I will not see this as a big problem but as something good to know!

another thing which I do not understand is how to solve this with a template? with a base class that define these operator maybe! but template?

My suggestion is to try to use TMAP from root instead of STL MAP, it is at least faster and then you are always sure that TObjects are inside so you do not care much about this!

regards

Mohammad

Subject: Re: std::map not treated properly
Posted by Bertram Kopf on Fri, 23 Jan 2009 16:27:59 GMT

View Forum Message <> Reply to Message

Hi Mohammad,

Quote:
Theoretically and in principle you are right that when std::map is used with pointers as keys the less operator should be implemented! and usually this is what you read in any STL book with the example of const char pointer and two pointer pointing to the same set of character and then the MAP can not decide it self which one is bigger! but here this is completely meaningless and will never introduce errors unless some body come to the idea to compare two tracks (e.g if (CbmMCTrack *t1 < CbmMCTrack *t2)) what ever this comparison means!

and even if we try to implement this less operator what does it mean here how can a track be less or larger than another?

I think the important thing here is the IsEqual, but this we have as we inherits from TObject so I will not see this as a big problem but as something good to know!

If one uses maps one "has to" guarantee that all functionalities of this template can be used properly. This means that it should be also possible to sort or to compaire things with the methods provided by std::map. Therefore it is mandatory to configure maps correctly. Otherwise it would be very error prone, especially in software projects where more than one developer is involved.

Quote:
another thing which I do not understand is how to solve this with a template? with a base class that define these operator maybe! but template?

The template could look like this:

```
struct PtrLess {
      template<class PtrType>
      bool operator()(PtrType ptr1, PtrType ptr2) const {
        return (*ptr1) < (*ptr2);
      }
    };
```

This requires that in each key object the less operator is reasonably defined.

Quote:
My suggestion is to try to use TMAP from root instead of STL MAP, it is at least faster and then you are always sure that TObjects are inside so you do not care much about this!

I am not sure but I think it is a question of tast to use TMap or std::map.

Cheers,
Bertram.