Subject: Template problem
Posted by bytefield on Wed, 20 Feb 2008 21:54:44 GMT
View Forum Message <> Reply to Message

Hi. I have one example about how to create a "safe" vector inheriting it from STL vector. The code is showed below. I have errors compiling it under Ubuntu.

```
#include <vector>
#include <iostream>
#include <string>
using namespace std;

struct Entry
{
 string name;
 int number;
};

template< class T > class Vec: public vector< T >
{
public:
 Vec(): vector< T >(){}
 Vec(int s): vector< T >(s){}

 T& operator[](int i) { return vector<T>::at(i); }
 const T& operator[](int i) const { return vector<T>::at(i); } // oups, forgotten to put last const
keyword
};

Vec< Entry > tel(1000);

void print_entry(int i)
{
 cout << tel[i].name << ' ' << tel[i].number << '\n';
}

int main()
{
 try
 {
  print_entry(1000);
 }
 catch(out_of_range)
 {
  std::cerr << "Domain error\n";
  return 1;
 }
 catch(...)
```

```
 {
  std::cerr << "Unknow exception\n";
  return 1;
 }
 return 0;
}
```

What could be the problem(s)?

Compiler version:

```
Using built-in specs.
Target: i486-linux-gnu
Configured with: ../src/configure -v --enable-languages=c,c++,fortran,objc,obj-c++,treelang
--prefix=/usr --enable-shared --with-system-zlib --libexecdir=/usr/lib --without-included-gettext
--enable-threads=posix --enable-nls --with-gxx-include-dir=/usr/include/c++/4.1.3
--program-suffix=-4.1 --enable-__cxa_atexit --enable-clocale=gnu --enable-libstdcxx-debug
--enable-mpfr --enable-checking=release i486-linux-gnu
Thread model: posix
gcc version 4.1.3 20070929 (prerelease) (Ubuntu 4.1.2-16ubuntu2)
```

Errors:

```
vec.cpp:22:   instantiated from here
```

L.E: See line 19 const T& operator[](int i) const { return vector<T>::at(i); }
My mistake, now overloading the [] operator work.

Subject: Re: Template problem
Posted by bytefield on Wed, 20 Feb 2008 22:13:26 GMT
View Forum Message <> Reply to Message

Strange...
In Windows with MVC 2005 EE it compiles well and work as expected if i comment out the second redefinition of [] operator. On linux(and Windows with mingw(g++)) still get errors about catch.

Subject: Re: Template problem
Posted by bytefield on Thu, 21 Feb 2008 09:59:26 GMT
View Forum Message <> Reply to Message

Problem solved...
Guess the problem is with g++ STL implementation. It is different from MS implementation. Maybe i give a try with stlport, before really closing the problem.
I also have some questions: which compiler you use daily?
which one appear to implement better the C++ standard?
on which platform you develop your software?

Subject: Re: Template problem
Posted by bytefield on Thu, 21 Feb 2008 10:45:40 GMT
View Forum Message <> Reply to Message

Problem closed. Stlport is almost alike g++ STL. I'm get tired trying to understand those headers. Never try to understand STL implementation! I mean it deep implementation. The surface always seems clear

Subject: Re: Template problem
Posted by masu on Thu, 21 Feb 2008 11:00:34 GMT
View Forum Message <> Reply to Message

Hi,

your problem rather is related to wrong function overloading.
If you want to overload a function with another one, they must differ in their signature, i.e. they cannot have the same parameter list. The return value is not part of the signature, so you get an error.

You should decide which version to take.
I don't know what the MSC does, but I think it silently discards one function (which in my opinion is not a good habit).

Matthias

Subject: Re: Template problem
Posted by bytefield on Thu, 21 Feb 2008 16:38:48 GMT
View Forum Message <> Reply to Message

OK. I know now about overloading function, that's way I've commented one out. But the problem still remain first catch (with out_of_range) and how out_of_range exception is raised in g++ STL implementation. In MS STL implementation (who in past was worse than g++ STL) it compile well. Seems g++ don't recognize such an exception. That's why I've asked which compiler others use. Expected type-specifier before 'out_of_range' means that out_of_range need to be of some type(and seems it's not). If it is Standard Template Library shouldn't it work in the same way on each platform or compiler which come with such implementation?
If out_of_range exception is implemented in MSC STL and even Bjarne Stroustrup (which guess use a unix os) spoken in his book(from where I've got this example) about it, shouldn't it be available on all STL impl.? (Well, guess Bjarne use his own version of compiler.)

Subject: Re: Template problem
Posted by masu on Fri, 22 Feb 2008 10:00:00 GMT
View Forum Message <> Reply to Message

Concerning the exception, I get the error:
d:\programs\upp-svn\MyApps\StdVecTst\StdVecTst.cpp: In function `int main()':
d:\programs\upp-svn\MyApps\StdVecTst\StdVecTst.cpp:38: error: `out_of_range' has not been declared
d:\programs\upp-svn\MyApps\StdVecTst\StdVecTst.cpp:38: error: invalid catch parameter
d:\programs\upp-svn\MyApps\StdVecTst\StdVecTst.cpp:38: error: `...' handler must be the last handler for its try block

After looking through c++ headers, I found I need to include 'stdexcept' and then it compiles fine. It is nowhere included in the other headers. I assume MSC includes it somewhere in vector headers and that's why there is no problem.

WinXP, Mingw 3.4.2

Matthias

Subject: Re: Template problem
Posted by bytefield on Fri, 22 Feb 2008 10:25:47 GMT
View Forum Message <> Reply to Message

Yes, you are right. It solve the problem. Thanks.

Subject: Re: Template problem
Posted by waxblood on Fri, 22 Feb 2008 13:32:20 GMT
View Forum Message <> Reply to Message

I remember Mirek once said somewhere in the forum that inheriting classes from upp containers is discouraged, as it is with STL... . Maybe problems arise just because STL is not designed to do such operations, but I can't tell more.

David

Subject: Re: Template problem
Posted by bytefield on Fri, 22 Feb 2008 13:58:47 GMT
View Forum Message <> Reply to Message

waxblood wrote on Fri, 22 February 2008 15:32I remember Mirek once said somewhere in the forum that inheriting classes from upp containers is discouraged, as it is with STL... . Maybe problems arise just because STL is not designed to do such operations, but I can't tell more.

David

Yeah i know it's not a good behavior but in that case(it was an example from Bjarne C++ programming book) std::vector doesn't raise any exception if you use [] operator over the vector limits and it could result in segmentation fault. So a solution to this problem is to inherit from vector and in new [] operator use return vector::at(i) which raise an exception if i is out of vector limits(see first example).

Sometimes is useful to inherit from containers[ex. if you want to make some gui stuff(ex. sizers) and cannot keep container as member].