Subject: Bug in msc 7.1 ?
Posted by mdelfede on Mon, 08 Mar 2010 21:55:42 GMT
View Forum Message <> Reply to Message

```
#include <Core/Core.h>

using namespace Upp;

struct pippo : Moveable<pippo>
{
 int T;

};

Vector<pippo> pluto()
{
 Vector<pippo> p;
 p.Add();
 return p;
}

CONSOLE_APP_MAIN
{
 Vector<pippo>v = pluto();

}
```

gives this error :

```
z:\home\massimo\WINDOWS\upp\uppsrc\Core\Topt.h(154) : error C2523: 'pippo::~T' : destructor
tag mismatch
        z:\home\massimo\WINDOWS\upp\uppsrc\Core\Vcont.hpp(68) : see reference to function
template instantiation 'void Upp::DestroyArray<T>(T *,const T *)'
 being compiled
        with
        [
            T=pippo
        ]
        Z:\home\massimo\WINDOWS\SDK2003\Microsoft Visual C++ Toolkit
2003\include\xmemory(136) : while compiling class-template member function 'void Upp::V
 ector<T>::Free(void)'
        with
        [
            T=pippo
        ]
```

Z:\home\massimo\WINDOWS\upp\MyApps\TestVectx\TestVectx.cpp(12) : see reference to class template instantiation 'Upp::Vector<T>' being compiled
with
[
T=pippo
]

Changing the definition of struct pippo like that :

```
struct pippo : Moveable<pippo>
{
 int _T;  // <== note, now variable is _T

};
```

Solves the problem... sigh. It seems to me that M$ compiler makes some mixing between template parameters and variable names in structs....
As usual, GCC behaves right there.

Max

Subject: Re: Bug in msc 7.1 ?
Posted by Lance on Sun, 01 Dec 2013 01:46:28 GMT
View Forum Message <> Reply to Message

Interesting. I remember the other day I encountered this problem:

```
extern "C" extern int array[10];
```

Compiles fine on VC++11 but will not be accepted in g++ 4.8

I have to revise it to something like

```
extern "C"{
extern int array[10];
}
```

I am not sure which is more standard-complying here though.

And I have another very interesting situation. Some predicate I passed to Upp::BiFindIndex(...) stops both VC++11 and g++ upto 4.8 with a compiling error of cannot find a match. And the only thing I need to do to fix it is insert a blank line (or subsequently remove it if there is one already),

F5 and it will pass. This symptom is same on both compilers. A recent update on vc++ has fixed the issue but g++ remains the same.

Here is what I do. It can be easily simulated:

```
inline bool compare_person(const RecordSet::Record& r, int person_id)
{
 return r[0].As<int32>()<person_id;
}

String TrialBalanceCtrl::FormattedPerson(int person_id)
{
 String s;
[b]
 int i=BinFindIndex(person, person_id, compare_person);
 //above line will fail g++4.8 from time to time
 //simply add or remove a line after it will make
 //g++4.8 happy on the next compile pass.
[/b]
 if(i!=person.RecordCount() && person(i,0).As<int32>()==person_id)
 {
  s<<'['<<person(i,"code").As<String>()
    <<"] "<<person(i,"name").As<String>();
 }
 return s;
}
```