Subject: Interesting struggle with "Moveable<T>" usage in GCC
Posted by mr_ped on Thu, 17 Jul 2008 13:13:59 GMT
View Forum Message <> Reply to Message

Hello, I did hit recently a compilation error I didn't understand fully, still I did find a workaround, so I'm fine, but maybe some compiler guru will be able to explain it better to me.

Let's consider (I always end with OOP design like this somewhere, I'm not sure whether it's a curse or good thing, but my mind loves constructs like this one):

```
class CT {
public:
  struct ST : Moveable<ST> {
    int x, y;
  };
};
//Nice little embedded struct inside a class.
//Perfectly Moveable too, just two ints.

//But now you want to initialize it like with classic struct:
/* 1) */     CT::ST a = { 0, 1 };
/* 2) */     CT::ST b[2] = { { 0, 1 }, { 3, 4 } };
```

During compilation with GCC (didn't try it with MSC, sorry) you get errors:
1) error: braces around initializer for non-aggregate type 'CT::ST'
2) error: braces around scalar initializer for type 'CT::ST'

I was like, what's wrong with simple struct initializer? As this really does save me lot of time while I'm writing unit tests, I decided to investigate it a bit more, and by hopeless trying to change every piece of that source I figured out how to make it work. The "correct" way to define such struct to be both Moveable, and allow you to write direct initialization with braces is like this:

```
//classic struct definition
class CT {
public:
  struct ST {
    int x, y;
  };
};

//make it also moveable, so NTL will store it in Vector container
namespace Upp {
  NTL_MOVEABLE( CT::ST );
}
```

(I tried to compile the "fixed" source under MSC, now it's complaining about missing DeepCopyConstruct ... fortunately after adding that one both compilers are happy (the manual page http://www.ultimatepp.org/srcdoc$Core$pick_$en-us.html did told me how to fix deep copy))

So, I have almost no idea why the compiler is such b*tch about basically the same source just written in different way, but in case somebody hits this problem, I'm posting my "work around".

Also I welcome any reasonable explanation, or eventually fix to "Moveable<T>" if possible (I think it's impossible, maybe due to template nature?), because it's much more neat to read in source ": Moveable<T>" than the NTL_MOVEABLE macro followed by 4 deepcopy functions in worst case surrounded by yet another namespace definition :/. I prefer when the source contains minimum of "accident" content, and looks almost like pure "essence" thing, if you know what I mean.