Subject: Re: Anonymous delegates
Posted by Zardos on Fri, 07 Dec 2007 10:07:23 GMT
View Forum Message <> Reply to Message

luzr wrote on Fri, 07 December 2007 09:36
Well, is not C++ fun?
Mirek

I'm a split personality about my C++ opinion.
On one day I think C++ is one of the most horrible languages ever invented.
And on another day I'm amazed and impressed about how much thought has been put into the language...

luzr wrote on Fri, 07 December 2007 09:36
Anyway, IMO this "foreach" has problem:

```
if(x)
   foreach(int a, v)
```

Mirek

Yes you are right! Thats not nice!
I currently can not test the code, but I think this should solve the problem:

```
struct IterHolder {
 void *p;
 void *x;

 IterHolder(bool) {}
 operator bool() const                          { return false; }

 template<class T> Begin(const T& v)              { p = (void*)v.Begin(); x = (void*)v.End(); }
 template<class T> End(const T& v)               { p = (void*)((v.End()) - 1); x = (void*)v.Begin(); }
 template<class T> Prev(Type2Type<T>)               { p = ((T*)p) - 1; }
 template<class T> Next(Type2Type<T>)               { p = ((T*)p) + 1; }
                bool CheckF() const             { return p < x; }
                bool CheckB() const             { return p >= x; }
 template<class T> T& Get(Type2Type<T>) const       { return *((T*)p); }

};

#define foreach(e, arr) \
 if(IterHolder _ith_ = false) {} else \
  for(_ith_.Begin(arr); _ith_.CheckF(); _ith_.Next(ENCODED_TYPEOF(arr[0]))) \
   if(bool _foreach_continue = true) \
    for(e = _ith_.Get(ENCODED_TYPEOF(arr[0])); _foreach_continue; _foreach_continue = false)
```

I'm not sure if the c++ optimizer can still remove all the noise and create a simple iterater loop for this version. But I guess performance should still be the same as a hand written loop.

Before using this code in production I probably would tweak the IterHolder and the Upp containers a little bit and make it more generic. I already have written 4 version of foreach and currently using a slightly different version, but I get tired of it... The concept is always the same. The main trick is ENCODED_TYPEOF(...) to get a the type of an expression without evaluating it.

- Ralf
