Subject: Re: destroying self from array
Posted by dolik.rce on Thu, 06 May 2010 13:14:04 GMT
View Forum Message <> Reply to Message

qwerty wrote on Thu, 06 May 2010 14:44the worst thing I experience is after namy hours of searchnig/experimenting is ask on the forum and few minutes later find the solution   Yes, almost as bad as when someone answers his own question just when you come up with the answer

Anyway, I would use similar approach, but based on the address of slave (didn't know they have id).
```
#include <Core/Core.h>
using namespace Upp;

class T;

class Master {
public:
 Array<T> slaves;
 void KillSlave(T* slave){
  for(int i = 0; i < slaves.GetCount(); i++){
   if(&(slaves[i])==slave) slaves.Remove(i);
  }
 }
};

class T {
public:
 Master* the_lord;
 T(Master* the_lord) : the_lord(the_lord) {}
 void kill_me() {
  the_lord->KillSlave(this);
 }
};


CONSOLE_APP_MAIN{
 Master m;
 m.slaves.Add(&m);
 m.slaves.Add(&m);
 DUMP(m.slaves.GetCount());
 m.slaves[0].kill_me();
 DUMP(m.slaves.GetCount());
}
```

But what puzzles me, is why would you need that? I have trouble imagining how you call kill_me() without knowing it's index. Te only reason I could come up with is that you use pointers to the slaves somehow. The cleanest way to do this might be using ArrayIndex. The problem would than have single line solution:
```
void kill_me() {
  the_lord->slaves.Remove(the_lord->slaves.Find(*this));
}
```

Regards,
Honza