Subject: Wrong(?) ASSERT in Vector<>::Insert()
Posted by dolik.rce on Wed, 04 Aug 2010 21:33:18 GMT
View Forum Message <> Reply to Message

Hello,
I just hit a problem in Vector code and I feel a need for explanation . Have a look at this code

```
#include <Core/Core.h>
using namespace Upp;

CONSOLE_APP_MAIN{
 Vector<int> v;
 v.Add(123);
 v.Insert(0,v[0]);
}
```

I would expect it to insert a deep copy of the first element of the Vector and place it before the original. The actual result is that the code hits an assert in the Insert function: template <class T>

```
void Vector<T>::Insert(int q, const T& x, int count) {
 if(!count) return;
 ASSERT(&x < vector || &x > vector + items);
 RawInsert(q, count);
 DeepCopyConstructFill(vector + q, vector + q + count, x);
}
```

As you can see, it check if the inserted data are part of the object itself. And of course, in the example above, they are.

Now, can someone explain me the reason for this assert? I fail to see how it could be harmful to make a deep copy of one of the elements inside...

If there is no real reason for this, I would like to see that assertion removed. There are also the same or similar assertions in the other variations of Insert which can be probably removed.

In my actual code I had to work around this by first making a deep copy of the copied element in temporary object and than inserting that. But I use quite big objects, not just int, so this workaround introduces is not only ugly, but also slow.

Best regards,
Honza