Subject: Using Vector::At doesn't initialize implicit types
Posted by cbpporter on Tue, 29 Jul 2008 08:51:02 GMT
View Forum Message <> Reply to Message

If I use At or SetLength, Vecor will call the default constructor for the new items created which are not manually initialized for user classes, but will leave implicit types uninitialized.

Example:

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

using namespace Upp;

struct Foo: public Moveable<Foo>
{
 int x;

 Foo()       { x = 0; }
 Foo(int xx)      { x = xx; }
};

CONSOLE_APP_MAIN
{
 Vector<int> v;
 v.At(0) = 7000;
 v.At(2) = 7000;

 DUMP(v[0]);
 DUMP(v[1]);
 DUMP(v[2]);

 Vector<Foo> w;
 w.At(0) = Foo(7000);
 w.At(2) = Foo(7000);

 DUMP(w[0].x);
 DUMP(w[1].x);
 DUMP(w[2].x);
}
```

I think we should change this. It can lead to problems and nasty crashes if you have a Vector<Foo>, where Foo is a value type class, and for some reason you need to start using Vector<Foo*>. If you Vector is not initialized in order, which is not a requirement for this class, you will most likely get memory corruption when you try to access the gaps and you cant test for NULL either.

Subject: Re: Using Vector::At doesn't initialize implicit types
Posted by mr_ped on Tue, 29 Jul 2008 08:52:06 GMT
View Forum Message <> Reply to Message

At(index, init_value);

edit:
BTW, I like current behavior.
The uninitialized memory bugs don't happen to me anymore (unit testing + valgrind), and for performance reasons the current behavior of At is better for me.

Subject: Re: Using Vector::At doesn't initialize implicit types
Posted by cbpporter on Tue, 29 Jul 2008 09:10:00 GMT
View Forum Message <> Reply to Message

Yes, using that At works if you want to initialize it there.

I was talking about the way user defined classes get their default constructor called, while implicit types get left initialized.
It should be more uniform.

And since user types do get their constructor called, I don't think there is any performance difference between what I'm proposing and the current situation.

Subject: Re: Using Vector::At doesn't initialize implicit types
Posted by mr_ped on Tue, 29 Jul 2008 09:13:58 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Tue, 29 July 2008 11:10Yes, using that At works if you want to initialize it there.

I was talking about the way user defined classes get their default constructor called, while implicit types get left initialized.
It should be more uniform.

And since user types do get their constructor called, I don't think there is any performance difference between what I'm proposing and the current situation.

No, it does fix your problem, try it.

```
...
 Vector<int> v;
 v.At(0, 0) = 7000;
 v.At(2, 0) = 7000;
...
```
LOG:

v[0] = 7000
v[1] = 0
v[2] = 7000

I'm using basic types a lot, and with vectors over 1mil in size, so unwanted initialization would slow me down considerably.

Actually I think calling the user defined constructor may be inconsistent behavior, and maybe it should not happen in your example.

edit:
And the "int" nature of vector size is starting to scary me, I would rather prefer full 32b size_t (although I understand the pick behavior would interfere much more in such scheme without additional flag (increasing the memory footprint of Vector more)) :/ This new HW in recent years makes 2+GB of data to look like an easy chew, yet SW like UPP::Vector will give you some headache in such case, and you will end up probably with custom containers anyway. On 64b system the int size of Vector will become problem in future.

Subject: Re: Using Vector::At doesn't initialize implicit types
Posted by cbpporter on Tue, 29 Jul 2008 09:20:18 GMT
View Forum Message <> Reply to Message

mr_ped wrote on Tue, 29 July 2008 12:13
No, it does fix your problem, try it.

I know that it fixes my problem!

I wasn't arguing about that problem, I was saying that the lack of uniformity between user and implicit types is a bad thing. We should have uniform semantics so we don't get unexpected security flaws.

Subject: Re: Using Vector::At doesn't initialize implicit types
Posted by mr_ped on Tue, 29 Jul 2008 09:22:04 GMT
View Forum Message <> Reply to Message

Ok, I see, but I vote not for initialization, but for not calling constructor on user types to make it consistent.

Subject: Re: Using Vector::At doesn't initialize implicit types
Posted by mirek on Tue, 29 Jul 2008 09:24:35 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Tue, 29 July 2008 05:10Yes, using that At works if you want to initialize it

there.

I was talking about the way user defined classes get their default constructor called, while implicit types get left initialized.
It should be more uniform.

And since user types do get their constructor called, I don't think there is any performance difference between what I'm proposing and the current situation.

Actually, this aspect of C++ is quite confusing. For any user type, creating a variable of that type without further params calls default constructor.

Not so for fundamentals. But you can "call default constructor" for them as well - which assigns zero to them.

Anyway, I think there is some small performance impact for At - assigning zero is not free. Usually, this would be negligible, but I can imagine several usages where it could have significant impact.

Mirek

Subject: Re: Using Vector::At doesn't initialize implicit types
Posted by cbpporter on Tue, 29 Jul 2008 14:03:09 GMT
View Forum Message <> Reply to Message

Well I guess we can leave it as is for performance reasons, but then we should add an entry to "U++ traps and pitfalls". This behavior should be documented somewhere, made very clear especially for pointers. Using pointers in Vector is not that common, so when you start using them it can be a nasty surprise that the values are not null checkable.

Subject: Re: Using Vector::At doesn't initialize implicit types
Posted by mirek on Tue, 29 Jul 2008 16:39:14 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Tue, 29 July 2008 10:03Well I guess we can leave it as is for performance reasons, but then we should add an entry to "U++ traps and pitfalls". This behavior should be documented somewhere, made very clear especially for pointers. Using pointers in Vector is not that common, so when you start using them it can be a nasty surprise that the values are not null checkable.

I do not know. If you do not initialize variable, you can consider it unitialized. If you do not initialize struct members, ditto.

What is surprising on At behaviour? Especially if it has another "init" method variant.

But OK, you are perhaps right.

Mirek

Subject: Re: Using Vector::At doesn't initialize implicit types
Posted by cbpporter on Tue, 29 Jul 2008 19:55:45 GMT
View Forum Message <> Reply to Message

luzr wrote on Tue, 29 July 2008 19:39
I do not know. If you do not initialize variable, you can consider it unitialized. If you do not initialize struct members, ditto.

I wouldn't say ditto, because if I do not initialize structs, they do get initialized for me automatically via default constructor, so theoretically they are initialized. If the constructor does not initialize properly that is a different problem .
