Subject: Proposal: add Vector::InsertPick(int i, pick_ T&)
Posted by Mindtraveller on Fri, 08 May 2009 07:48:49 GMT
View Forum Message <> Reply to Message

Let`s imagine we have some Moveable class M. And we make it`s vector with Vector<M>. So it is possible to use it without any additional constructors if we add new items with Vector<M>::AddPick. But when we need to add something somewhere inside vector and try to call Vector<M>::Insert, compiler requires M to have deep copy ctor. Actually it is not really necessary to have deep copy in this case, as picking would be sufficient. Isn`t it? So I propose adding InsertPick(int i, pick_ T&).

Subject: Re: Proposal: add Vector::InsertPick(int i, pick_ T&)
Posted by mrjt on Fri, 08 May 2009 08:07:32 GMT
View Forum Message <> Reply to Message

Well, perhaps it should be added for symmetry, but doesn't this:

Vector<M> v;
M a;
..
v.Insert(0) = a;

Do exactly the same thing?

Subject: Re: Proposal: add Vector::InsertPick(int i, pick_ T&)
Posted by Mindtraveller on Fri, 08 May 2009 08:30:29 GMT
View Forum Message <> Reply to Message

Actually it doesn`t. Differences are:
- you require M to have default constructor, which is not strong requirement for Vector element
- you have one redundant call (ctor and operator= instead of ctor), which sometimes is crucial (picking was made for efficiency, right?)

And IMO this code is little bit less clear than plain v.InsertPick(a).

Subject: Re: Proposal: add Vector::InsertPick(int i, pick_ T&)
Posted by mirek on Fri, 08 May 2009 19:21:30 GMT
View Forum Message <> Reply to Message

Mindtraveller wrote on Fri, 08 May 2009 04:30Actually it doesn`t. Differences are:
- you require M to have default constructor, which is not strong requirement for Vector element

Does it?

Quote:
- you have one redundant call (ctor and operator= instead of ctor), which sometimes is crucial (picking was made for efficiency, right?)

And IMO this code is little bit less clear than plain v.InsertPick(a).

While I can agree with symmetry point and clarity issue, I do not see how adding InsertPick should solve above problem.

Maybe you can rewrite mrjt's example with InsertPick?

Mirek

Subject: Re: Proposal: add Vector::InsertPick(int i, pick_ T&)
Posted by Mindtraveller on Fri, 08 May 2009 22:05:54 GMT
View Forum Message <> Reply to Message

luzr wrote on Fri, 08 May 2009 23:21Mindtraveller wrote on Fri, 08 May 2009 04:30Actually it doesn`t. Differences are:
- you require M to have default constructor, which is not strong requirement for Vector element

1. Does it?

Quote:
- you have one redundant call (ctor and operator= instead of ctor), which sometimes is crucial (picking was made for efficiency, right?)

And IMO this code is little bit less clear than plain v.InsertPick(a).

2. While I can agree with symmetry point and clarity issue, I do not see how adding InsertPick should solve above problem.

3. Maybe you can rewrite mrjt's example with InsertPick?

Mirek
1. I`m not quite shure I understand what you mean by this question.
If I understand correctly and the question is about default ctor requirement, TheIDE help says:
Quote: General requirement: T is required to be moveable and must have either deep copy constructor, pick constructor or default constructor.

Adds new element to Vector and picks value of parameter to it.
Requires T to have pick constructor.
-- so it is not necessary for M to have default /deep copy ctor to have Vector<M> and do AddPick.

2. I hope that InsertPick won`t make me writing default constructors or optional deep copy to insert element into queue. This should solve the problem with deep copy where I want to pick only.

3.Vector<M> v;
M a;
..
v.InsertPick(a);
Or maybe you mean InsertPick(int i, pick_ Vector<M> &) ?

Subject: Re: Proposal: add Vector::InsertPick(int i, pick_ T&)
Posted by mirek on Sat, 09 May 2009 07:02:30 GMT
View Forum Message <> Reply to Message

Yes, my mistake, you are right.

I will add InsertPick ASAP.

Mirek

Subject: Re: Proposal: add Vector::InsertPick(int i, pick_ T&)
Posted by mirek on Sat, 09 May 2009 07:06:51 GMT
View Forum Message <> Reply to Message

This should do the trick:

```
template <class T>
void Vector<T>::InsertPick(int i, pick_ T& x)
{
 if(!count) return;
 ASSERT(&x < vector || &x > vector + items);
 RawInsert(q, count);
 ::new(vector[q]) T(x);
}

template <class T>
void Array<T>::InsertPick(int i, pick_ T& x)
{
 vector.InsertN(i, 1);
 vector[i] = new T(x);
```

}

Mirek

Subject: Re: Proposal: add Vector::InsertPick(int i, pick_ T&)
Posted by Mindtraveller on Sat, 09 May 2009 19:52:50 GMT
View Forum Message <> Reply to Message

Thank you very much! U++ is great at it`s support.

Subject: Re: Proposal: add Vector::InsertPick(int i, pick_ T&)
Posted by hans on Sun, 10 May 2009 17:56:19 GMT
View Forum Message <> Reply to Message

Hi,
I think

```
template <class T>
void Vector<T>::InsertPick(int i, pick_ T& x)
{
 if(!count) return;
 ASSERT(&x < vector || &x > vector + items);
 RawInsert(q, count);
 ::new(vector[q]) T(x);
}
```

is wrong, should be:

```
template <class T>
void Vector<T>::InsertPick(int q, pick_ T& x)
{
 ASSERT(&x < vector || &x > vector + items);
 RawInsert(q, 1);
 ::new(vector[q]) T(x);
}
```

Thanks, Hans.

Subject: Re: Proposal: add Vector::InsertPick(int i, pick_ T&)
Posted by mirek on Sun, 10 May 2009 21:11:23 GMT

hans wrote on Sun, 10 May 2009 13:56Hi,
I think

```
template <class T>
void Vector<T>::InsertPick(int i, pick_ T& x)
{
 if(!count) return;
 ASSERT(&x < vector || &x > vector + items);
 RawInsert(q, count);
 ::new(vector[q]) T(x);
}
```

is wrong, should be:

```
template <class T>
void Vector<T>::InsertPick(int q, pick_ T& x)
{
 ASSERT(&x < vector || &x > vector + items);
 RawInsert(q, 1);
 ::new(vector[q]) T(x);
}
```

Thanks, Hans.

Thank you, you are right, of course.

Mirek