Subject: Heap-leaks and polymorphic containers
Posted by mrjt on Mon, 12 May 2008 15:41:09 GMT
View Forum Message <> Reply to Message

It is my understanding that the following code should execute without any memory leaks:

```
struct Item {
 String name;
};

struct Collection : public Item
{
 Array<Item> items;
};

GUI_APP_MAIN
{
	// Example 1
 Array<Item> array;
 array.Create<Collection>().items.Add();

	// Example 2
 Collection *col = new Collection();
 col->items.Add();
 Item *item = (Item *)col;
 delete item;
}
```

But for some reason the destructor for 'items' isn't getting called. Adding a virtual destructor to force clearance just causes a crash in MemoryFreeDebug.

Is something broken or am I just missing something obvious?

Subject: Re: Heap-leaks and polymorphic containers
Posted by mirek on Mon, 12 May 2008 17:05:28 GMT
View Forum Message <> Reply to Message

mrjt wrote on Mon, 12 May 2008 11:41It is my understanding that the following code should execute without any memory leaks:

```
struct Item {
 String name;
};

struct Collection : public Item
{
 Array<Item> items;
};

GUI_APP_MAIN
```

```
{
	// Example 1
 Array<Item> array;
 array.Create<Collection>().items.Add();

	// Example 2
 Collection *col = new Collection();
 col->items.Add();
 Item *item = (Item *)col;
 delete item;
}
```
But for some reason the destructor for 'items' isn't getting called. Adding a virtual destructor to force clearance just causes a crash in MemoryFreeDebug.

Is something broken or am I just missing something obvious?

Well, virtual destructor is absolutely required here.

The crash in MemoryFreeDebug... well, who knows, I would say it has another reason.

Mirek

Subject: Re: Heap-leaks and polymorphic containers
Posted by mrjt on Tue, 13 May 2008 13:46:55 GMT
View Forum Message <> Reply to Message

luzr wrote on Mon, 12 May 2008 18:05Well, virtual destructor is absolutely required here.Hmm. I understand why this works (the implicit destructor isn't virtual):
```
struct Item : public Moveable<Item> {
 String name;

 virtual ~Item() { }
};

struct Collection : public Item
{
 Array<Item> items;
};

GUI_APP_MAIN
{
 Collection *col = new Collection();
 col->items.Add();
 Item *item = (Item *)col;
 delete item;
}
```
But what baffles me is why this works with Vector, but not Array (the Array's destructor doesn't get

called but the Vector one does).

```
struct Item : public Moveable<Item> {
 String name;
};

struct Collection : public Item
{
 Vector<Item> items; // Heap leak if changed to Array
};

GUI_APP_MAIN
{
 Collection *col = new Collection();
 col->items.Add();
 Item *item = (Item *)col;
 delete item;
}
```

Subject: Re: Heap-leaks and polymorphic containers
Posted by mirek on Tue, 13 May 2008 14:00:21 GMT
View Forum Message <> Reply to Message

Arrrgh, a bug in leak detector...

Vector is using "MemoryAllocSz" variant, which is not implement with leak checking.

Mirek

Subject: Re: Heap-leaks and polymorphic containers
Posted by mrjt on Tue, 13 May 2008 14:02:27 GMT
View Forum Message <> Reply to Message

Thank god! I was beginning to think I was delusional

Subject: Re: Heap-leaks and polymorphic containers
Posted by mdelfede on Tue, 13 May 2008 21:39:17 GMT
View Forum Message <> Reply to Message

luzr wrote on Tue, 13 May 2008 16:00Arrrgh, a bug in leak detector...

Vector is using "MemoryAllocSz" variant, which is not implement with leak checking.

Mirek

That one drove me crazy when I was looking for memory violation bug !

Max

Subject: Re: Heap-leaks and polymorphic containers
Posted by mirek on Wed, 14 May 2008 06:30:06 GMT
View Forum Message <> Reply to Message

mdelfede wrote on Tue, 13 May 2008 17:39luzr wrote on Tue, 13 May 2008 16:00Arrrgh, a bug in leak detector...

Vector is using "MemoryAllocSz" variant, which is not implement with leak checking.

Mirek

That one drove me crazy when I was looking for memory violation bug !

Max

Yeah, a time to fix this.

Mirek
