

---

Subject: Fixes to Array::Create & Vector::Create  
Posted by [Novo](#) on Fri, 15 Dec 2017 02:46:27 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

Hi Mirek,

Could you please change methods Array::Create & Vector::Create to make them look like below?  
template<class TT, class... Args>

```
TT& Create(Args&&... args) { TT *q = new TT(pick(args)...); Add(q); return *q; }
```

template <class... Args>

```
T& Create(Args&&... args) { if(items >= alloc) GrowF(); return *(:new(Rdd())  
T(pick(args)...)); }
```

This shouldn't break anything, and as a bonus this should allow to pass arguments by reference in case they do not have a copy constructor, what is quite common with Upp.

Regards,

---

---

---

Subject: Re: Fixes to Array::Create & Vector::Create

Posted by [mirek](#) on Fri, 15 Dec 2017 08:10:28 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

Novo wrote on Fri, 15 December 2017 03:46Hi Mirek,

Could you please change methods Array::Create & Vector::Create to make them look like below?  
template<class TT, class... Args>

```
TT& Create(Args&&... args) { TT *q = new TT(pick(args)...); Add(q); return *q; }
```

template <class... Args>

```
T& Create(Args&&... args) { if(items >= alloc) GrowF(); return *(:new(Rdd())  
T(pick(args)...)); }
```

This shouldn't break anything, and as a bonus this should allow to pass arguments by reference in case they do not have a copy constructor, what is quite common with Upp.

Regards,

Just changing them to r-value would potentially break a lot. However, I am trying to add them as overload.

---

---

Subject: Re: Fixes to Array::Create & Vector::Create

Posted by [mirek](#) on Fri, 15 Dec 2017 08:28:33 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

```
template <class... Args>
T& Create(const Args&... args) { if(items >= alloc) GrowF(); return *(:new(Rdd()) T(args...)); }
template <class... Args>
T& Create(Args&&... args) { if(items >= alloc) GrowF(); return *(:new(Rdd()) T(pick(args...))); }
}
```

---

---

Subject: Re: Fixes to Array::Create & Vector::Create

Posted by [mirek](#) on Fri, 15 Dec 2017 08:44:53 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

Unfortunately, this seems not to work:

```
struct Test {
    Vector<int> a;
    int      b;

    Test(Vector<int>&& a, int b) : a(pick(a)), b(b) {}
};
```

```
CONSOLE_APP_MAIN
{
    Array<Test> h;
    Vector<int> v;
    v.Add(12);
    h.Create<Test>(v, 22);
    DDUMP(v.GetCount());
    h.Create<Test>(pick(v), 22);
    DDUMP(v.GetCount());
}
```

Here, I would expect const Args& for the first 'Create' and && for the second one.

For some reason, both end in && overload (I am really not sure why, IMO they should not, but they do with both MSC and GCC).

So unfortunately, to stay safe, I am going to rollback this feature. This has really trivial workaround (default constructor + Set method).

Mirek

---

---

Subject: Re: Fixes to Array::Create & Vector::Create  
Posted by [Novo](#) on Fri, 15 Dec 2017 16:40:49 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

Sorry, my bad. The code should look like below.

```
template<class TT, class... Args>
TT& Create(Args&&... args) { TT *q = new TT(std::forward<Args>(args)...); Add(q); return
*q; }

struct Test {
Test(const Vector<int>& a, int b) : a(a, 0), b(b) {}
Test(Vector<int>&& a, int b) : a(pick(a)), b(b) {}

Vector<int> a;
int      b;
};

CONSOLE_APP_MAIN
{
Array<Test> h;
Vector<int> v;
v.Add(12);
// Copy-constructor
h.Create<Test>(v, 22);
DDUMP(v.GetCount());
// Move-constructor
h.Create<Test>(pick(v), 22);
DDUMP(v.GetCount());
v.Add(21);
h.Create<Test>(Vector<int>(v, 0), 22);
DDUMP(v.GetCount());
}
```

I recompiled ThelDE with this change and it works perfectly for me.

How it works.

`type&&` is called a universal/perfect/forwarding reference and it can be either `type&&` or `type&` depending on arguments. This is why your method `Create(const Args&... args)` was redundant. `Create(Args&&... args)` is a better match.

The problem was that I was using `pick`, which unconditionally converts type to an rvalue, instead of `std::forward`, which preserves type (an lvalue will stay the lvalue).

---

---

Subject: Re: Fixes to Array::Create & Vector::Create  
Posted by [mirek](#) on Sat, 16 Dec 2017 08:24:29 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

Perfect, applied.

Now I have just to add this to all Create methods everywhere :) (not only Array/Vector)

---

---

---

---

Subject: Re: Fixes to Array::Create & Vector::Create

Posted by [Novo](#) on Sat, 16 Dec 2017 20:02:55 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

Thanks a lot!

---

---