Subject: HELPER: Value grouping to ValueArray
Posted by kohait00 on Thu, 18 Nov 2010 09:10:14 GMT
View Forum Message <> Reply to Message

hi all,

here comes a ArrayCtrl like grouper of Values to a ValueArray, returned as Value itself.
maybe its of help for someone..
since a variable arguments list is suboptimal in c++, this is definitely a better solution in general.
good work who ever did this for ArrayCtrl..

syntax:

Value ToValueArray(const Value& [, const Value&]...);

```
inline Value ToValueArray(const Vector<Value>& v)
{
 return ValueArray(Vector<Value>(v, 0));
}

inline Value ToValueArray(Vector<Value>& v)
{
 return ValueArray(v);
}

//$-Value ToValueArray(const Value& [, const Value& ]...);
#define  E__ToValueArray(I)      Value ToValueArray(__List##I(E__Value));
 __Expand(E__ToValueArray)
#undef   E__ToValueArray

#define E__Addv(I)    v << p##I
#define E__ToValueArrayF(I) \
Value ToValueArray(__List##I(E__Value)) { \
 Vector<Value> v; \
 __List##I(E__Addv); \
 return ValueArray(v); \
}
__Expand(E__ToValueArrayF)

CONSOLE_APP_MAIN
{
 Value v = ToValueArray(123, "Hallo", 23.0);
 RLOG(v);
}
```

File Attachments

1) ValueGroup.rar, downloaded 204 times

Subject: Re: HELPER: Value grouping to ValueArray
Posted by mirek on Sat, 20 Nov 2010 17:28:42 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Thu, 18 November 2010 04:10hi all,

here comes a ArrayCtrl like grouper of Values to a ValueArray, returned as Value itself.

What about doing this as constructor overloads? Perhaps starting with 2 parameters to avoid copy variant. Or is it too ambiguous?

Quote:
since a variable arguments list is suboptimal in c++, this is definitely a better solution in general.

That is why Format uses the same technique...

Quote:
good work who ever did this for ArrayCtrl..

Thanks

Mirek

Subject: Re: HELPER: Value grouping to ValueArray
Posted by kohait00 on Sat, 20 Nov 2010 18:48:12 GMT
View Forum Message <> Reply to Message

What about doing this as constructor overloads? Perhaps starting with 2 parameters to avoid copy variant. Or is it too ambiguous?

as far as got to know the ValueArray code, it does adding values by adding it to internal Vector<Value> anyway.
doing it like in the example above already provides the prepared Vector<Value> to pick, so no copy is done. and it saves the ValueArray interface from growing and remains optional usage.

ambiguity maybe could be a side effect some day in other occassions.. i consider ValueArray(_pick Vector<Value>&) flexible enough if you ask me..

it'd also be possible to leave it like it is now and simply do

```
Value v = ValueArray(Vector<Value>() << 123 << "Hallo" << 34.9);
```

its not so nice, but neither generates extra code in Upp base to manage

it was just an idea, to increase readability

Subject: Re: HELPER: Value grouping to ValueArray
Posted by kohait00 on Thu, 02 Dec 2010 12:44:44 GMT

View Forum Message <> Reply to Message

i came to the conclusion that easy is best again

```
#define ASVALUEARRAY(x) ValueArray(Vector<Value>() << x)

Value v = ASVALUEARRAY(123 << "Hallo" << 34.9);
```

is actually same approach as in concatenating to cout, or in LOG()
so is almost natural
maybe this can go to Value.h..