GROMACS - Task #2925

BasicVector addition operator yields unexpected result when adding scalar

04/12/2019 04:42 PM - Christian Blau

Status:	New	
Priority:	Normal	
Assignee:	Christian Blau	
Category:	core library	
Target version:	2021-infrastructure-stable	
Difficulty:	uncategorized	

Description

The following code compiles, but gives an unexpected garbage result

```
RVec x{1,2,3};
    RVec y = x+2;
```

This works for all BasicVector types.

The likely reason for this is that the

- plus operator takes another BasicVector as argument ... operator+(const BasicVector & other)
- this is implicitely created from RawArray ... BasicVector(const RawArray x)
- the RawArray is implicitely converted from the single value (real x[DIM]{2})

A fix might be adding a scalar addition that matches before the operator+(const BasicVector & other).

History

#1 - 04/15/2019 12:48 PM - Roland Schulz

Pretty sure that isn't the reason. It can't use the operator+(const BasicVector & other) for two reasons:

1. a single value can't be converted to a C-style array, and
2. C++ doesn't do transitive conversions (if A is implicit convertible to B and B is to C, you can't implicit convert A to C).

I think the reason is:

- BasicVector gets converted to RawArray which is really just ValueType*.
- ValueType* + int is a pointer addition.

#2 - 04/15/2019 03:59 PM - Christian Blau

Roland Schulz wrote:

> I think the reason is:
>
> - BasicVector gets converted to RawArray which is really just ValueType*.
> - ValueType* + int is a pointer addition.

Agreed, also explains why RVec y = RVec{} + 2; compiles but not RVec y = RVec{} + 2.0;

#3 - 04/15/2019 04:15 PM - Mark Abraham

Given that explicit code is usually preferable to implicit code, I would like to see people write

```
RVec a{1, 2, 3};
real b{3};
RVec c = a + RVec{b, b, b};
// or
RVec d = a + realToUniformRVec(b);
```

I would hope that we can stop such bugs by declaring free functions for arithmetic of RVec and well chosen scalars, providing no definitions (so the code can't compile), and providing a general comment about why we have found this necessary. (Caveat, I have not tried to do this.)

#4 - 04/15/2019 10:18 PM - Mark Abraham

We should also consider removing the convenience conversion of BasicVector<ValueType> to ValueType[3], which would avoid the problem entirely. An as_rvec() member is also useful from the "prefer explicit" point of view

#5 - 04/20/2019 05:39 PM - Eric Irrgang

Mark Abraham wrote:

> We should also consider removing the convenience conversion of BasicVector<ValueType> to ValueType[3], which would avoid the problem entirely. An as_rvec() member is also useful from the "prefer explicit" point of view

That sounds good, but time consuming.

#6 - 12/17/2019 10:04 AM - Christian Blau

- Tracker changed from Bug to Task

- Target version changed from 2020 to 2021-infrastructure-stable

- Affected version deleted (N/A)