vec *C++ class for 3D vectors*

v2.1(β) / 07.03.2022

Mihai-Tiberiu Dima, Maria Dima
Meshcheryakov Lab. Information Tek, JINR, Russia

Header: `#include "vec.hh"`

Libraries: `libvec.a` (static)
`libvec.so` (dynamic)

linux-2.6.32-504.12.2.el6.x86_64
gcc-4.4.7 20120313

Variables (public):

- `x, y, z = scalar` .... polymorphic elements

Constructors (public)

- `vec<double> a` .... initialisation to zero, or a
- `vec<int> a(1,0,2)` .... initialisation to given value
- `auto a = b` .... initialisation to b (vec, or scalar)
- `auto a(b)` .... initialisation to b (vec, or scalar)

Assign (public)

- `vec<int> a; a = b` ....initialisation to b (vec, or scalar)

Cast operators

- `scalar(z)` .... conversion vec<scalar'> $\rightarrow$ scalar
- `vec<scalar>(z)` .... conversion vec<scalar'> $\rightarrow$ vec<scalar>

Negative (friend)

- `auto a = -b` ....initialisation to -b (const&, or &&)

Conjugate (friend)

- `auto a = ~b` ....initialisation to <b| (const&, or &&)

Algebraic operators (friends)

- `auto a = b+c` ....b, c are (const&, or &&), (vec)
- `auto a = b-c` ....b, c are (const&, or &&), (vec)
- `auto a = b*c` ....b, c are (const&, or &&), (vec, or scalar)
- `auto a = b/c` ....c is (const&, or &&), (scalar)
- `auto a += b` ....b is (const&, or &&), (vec)
- `auto a -= b` ....b is (const&, or &&), (vec)
- `auto a *= b` ....b is (const&, or &&), (vec, or scalar)
- `auto a /= b` ....b is (const&, or &&), (scalar)

- `auto a = (P|Q)` .... scalar-prod of vec P and Q

Functions (friend)

- `fabs(z)` .... norm

Print (friend)

- `cout << z << endl;` .... note 2 endl !
- `cout << boolalpha << z << endl;` ...... print scalar type appended

Usage examples

- `auto z = ~v` .... equal to <v|

Description

The ***vec*** class is a very slim (2 variables, constructors, cast operators) templated C++ class. The huge number of non-class operators (2400) are `friend`, saving an extra variable (`this`) in the call, for somewhat higher runtime expediency. A deeper reason is due to templated coding, each operator function needing ca. 7 implementations, in order to accomodate *quasi-polymorphism*.

Quasi-polymorphism means the package mimics polymorphism for the usual scalar types used in science and engineering. Statements such as:

```
auto z = double(1) * vec<int>(4,1,0) ;
```

benefit of the templated function type-calculator to determine the output type as `vec<double>.`

The class overloads `fabs` to calculate the norm – and has eigen to output a cpx<scalar> matrix w/ normed eigen-vec's as columns. Log and exp also available.

The class comes with all instantiation combinations for `int, float, double, long double – and cpx<int>, cpx<float>, cpx<double>, cpx<long double>`.

The ***makefile*** is banale, however with full pfledged functionality: `make libs`, `make test`, `make run`, `make clean`.

The class comes with 4 examples and 1 application example.