operators (lab9 cont'd)

Friday, October 15, 2010
9:00 AM

- for assignment operator

```
Vec_t& operator=(const Vec_t& rhs)
{
    if (this != &rhs) {
        // copy rhs
        for (int i=0; i<3; i++)
            vec[i] = rhs[i];
    }
    return *this;
}
```

don't forget this —
I forgot it in data_t (lab8)

Friday, October 15, 2010
9:06 AM

we want: where vec_t v1, v3, v4;

```
v4 = v1 - v3;

v4.operator=( v1.operator-(v3)) ;
```

↑ in vector.h;

```
// operator
vec_t operator-(const vec_t& rhs)
{
    vec_t result;   // call vec_t constructor — make sure constructor zero's out (initializes) this new instance
    for (int i=0; i<3; i++)
        result[i] = vec[i] - rhs[i];
    return result;
}
```

get copied into return type

```
std::cout << "v4 * 5 = " << v4 * 5 << std::endl;
std::cout << "3 * v4 = " << 3 * v4 << std::endl;

v4.operator*(double)

double.operator(vec_t)
(
```

```
friend vec_t operator*(const double& s, const vec_t& rhs)
{
    vec_t result;
    for(int i=0; i<3; i++)
        result[i] = s * rhs[i];
    return result;
}

friend vec_t operator*(const vec_t& lhs, const double& s)
{
    ;
}
```

---

another form of operator : unary

```
std::cout << "-v5 = " << -v5 << std::endl;
```

↑ unary − operator

```
vec_t operator-()
{
    vec_t result;
    for(int i=0; i<3; i++)
        result[i] = -vec[i];
```

return result;
}

;

// friends ostream
istream

;

// members
double dot(const vec_t&); } defined in implementation vector.cpp
double len();

private:
double vec[3]; what our vec_t actually is: a double array

}; // ends the class vec_t definition

---

in implementation (vector.cpp)

#include <iostream>
#include <iomanip> // C++ I/O manipulators
#include <cmath> => formatted output
e.g. << setw(5) ← 5.2
<< setprecision(2)
||| ✓
math.h

double vec_t::dot(const vec_t& rhs)
{

double vec_t::dot(const vec_t& rhs) const

{

double s=0.0;

for (int i=0; i<3; i++)

s += vec(i) * rhs[i];

return s;

}

lets me access all private members of vec_t, namely vec[] array

same as this->vec(i)

(*this).vec(i)

↑ reserved keyword for ptr to current object

double vec_t::len()

{

return (sqrt(dot(*this)));

}

↑

(*this).dot(*this)

---

pixel.h, pixel.cpp :

drgb_t

irgb_t

holds both

```
class drgb_t
{
    : // like vec_t
  private:
    double pix[3];   // otherwise pretty much the same as vec_t
};

class irgb_t
{
    : // like drgb_t
  private:
    unsigned char pix[3];
```

unsigned char p[4];

};

- you'll want to provide:
  - constructors
  - copy constructor
  - destructor (default ok)
    ~rgb_t() { };
  - assignment operator
  - operator[] accessor/mutator
  - operator +
  - operator *
  - friend operator *
  - members:
    bool iszero(void) const;

- repeat for rgb_t