C++ vec_t class

Wednesday, October 13, 2010
9:00 AM

- task: redo ray caster in C++
- lab 8: list_t & data_t

make sure you have the "big three":

- constructor
- copy constructor (object(rhs)) ← the class
- assignment operator (=)
- destructor, if needed (if constructor uses "new")

- today: vec_t class
  lab 9: old lab 9 + lab 10 combined

- motivation:

#include <iostream>
#include "vector.h"
↑
our c++ vect interface

```
int main()
{
    vec_t v1(1.0,2.0,3.0);
    vect v2(4.0,5.0,6.0);
    vect v3, v4, v5(v2);
```

this calls constructor

this calls compiler

vec_t(int ix=0.0, int iy=0.0, int iz=0.0)
4.0 5.0 6.0
{ vec[0]=ix; vec[1]=iy; vec[2]=iz; }

same fun name ⇒ overloading

calls copy constructor

v2

vec_t(const vec_t& rhs)
{ for(int i=0; i<3; i++)
    vec[i]=rhs[i];
}

double vec_t::operator[](int i)
{ return vec[i]; }

should really do bound checking
if(i>=0 && i<3)

still in main

std::cout << "Enter 3 vals" << std::endl;
std::cin >> v3;

friend std::istream& operator>>(std::istream& s, vec_t& rhs)

we let operator>> call vec_t class

we are overloading the std::istream operator>>

means we don't do a deep copy of rhs into the

we let istream.operator>>( ) get at internals of vec_t } / vec_t class

the std::istream. operator — not our object

a deep copy of rhs into file

s >> rhs[0] >> rhs[1] >> rhs[2];
return s; }

need this: vec_t::operator[ ]

in the vector.h header

double& operator[ ](int i) { return vec[i]; }

saw & can alter the type } MUTATOR of i^{th} element of our vec_t

⇓

vec[0] = 5.3;

what about double val = vec[1]; — ACCESSOR OF vec_t

overload operator[ ]

& don't want value changed. (const)

const double& operator[ ](int i) const { return vec[i]; }

back to main:

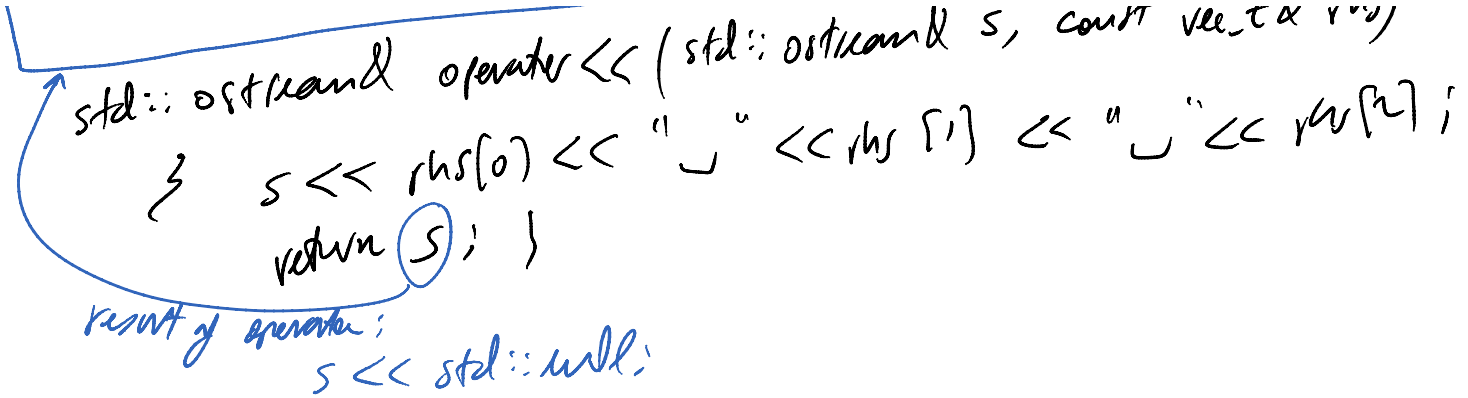
std::cout << "v1 = ␣" << v1 << std::endl;

same as "\n"

ostream operator<<

ostream& operator<<(std::ostream& s, const vec_t& rhs)

std::ostream& operator<<(



back in main:

v2[1] = 3.0;
v2[0] = v2[2];
} exercises our operator[]'s (both of them)

v4 = v1 - v3; → to get this to work, need to write

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

operator-( )
operator=( )

if you can do this, write

operator+

operator+