Monday, November 15, 2010
8:54 AM

Last time:

- STL containers, iterators; declarations:

```
#include <list>     std::list<int> ilist;
                    std::list<int>::iterator itr;

#include <vector>   std::vector<double> dvec;
                    std::vector<double>::iterator ditr;
```

- usage:

```
for (itr = ilist.begin(); itr != ilist.end(); ++itr)
    std::cout << *itr;
```

- and also: (eventually) (something like this)

```
#include <algorithm>
    sort (ilist.begin(), ilist.end());
```

} STL sort

so long as your list & objects on list support operator<
anything works

[illegible]

if this is a list, then sort could be insertion

if this is a vector, then sort could be quicksort

(provides operator[], direct access

std::vector<double> dlist;

dlist[0] = ... } ok for vector, not ok for list (I think)

Monday, November 15, 2010
9:15 AM

- what we want to do:

```
list_t <data_t *>                    list;

list_t <data_t *>::const_iterator    citr;
list_t <data_t *>::iterator          litr;

data_t                               *data = NULL;
std::string                          name;
int                                  num;

std::cout << "*** printing list ***" << std::endl;
for (citr = list.begin(); citr != list.end(); ++citr)
    std::cout << *citr;
```

① operator<< (data_t *rhs)
for data_t type
↑ pointer to data_t

assignment operator =
for object list_t<data_t*>::const_iterator

LHS operator++ for object

②

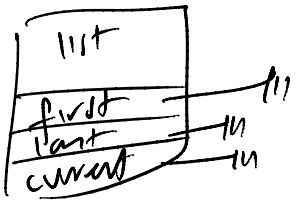
(2) need operator != test fcn.
for list_t<>::const_iterator

(3) for list_t<> list object: begin(), end() functions that return list_t<>::const_iterator

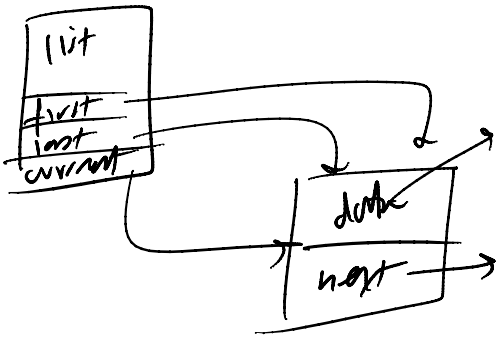
makes sense to define this as part of list itself.

Monday, November 15, 2010
9:27 AM

- old list:



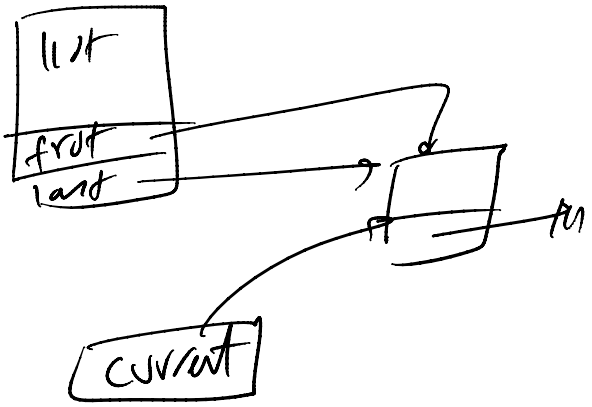
upon instantiation

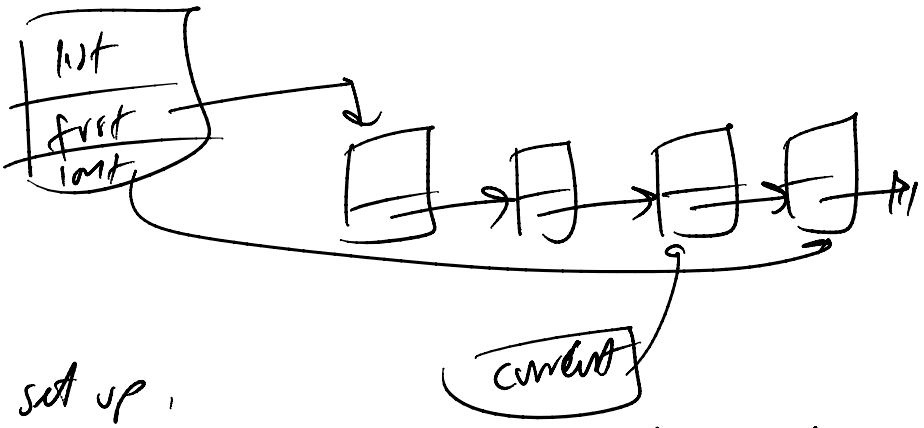


with one node on list

- idea: redo the list so that current ptr acts like an iterator

- think of list like this:





with this set up,

- can you write begin(), end() functions so that list iterator works?

e.g. `while( current = list.begin(); current != list.end(); ++current)`
`std::cout << *current;`

current is going to become our iterator

- how does list iterator work currently?

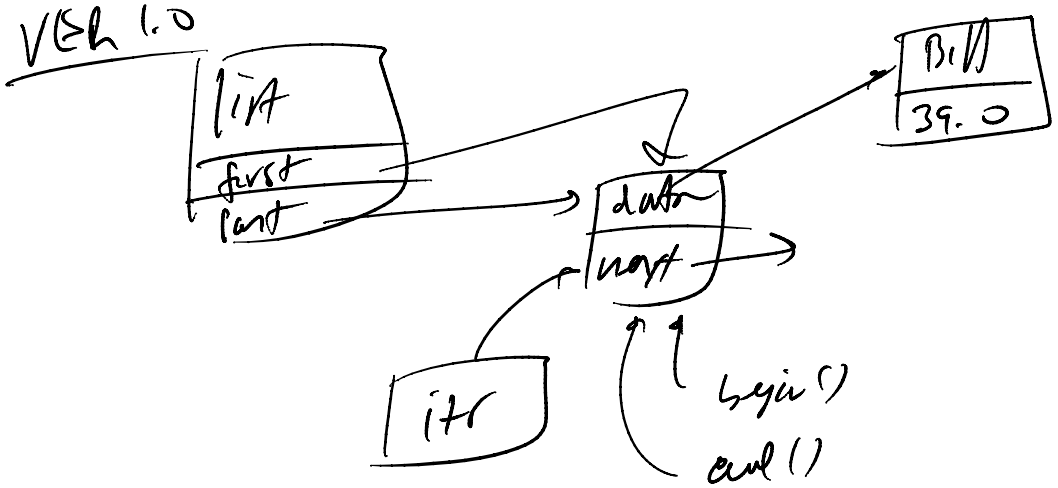
`list.reset();` — like `begin()`

`while (list.not_end()) {` — like `current != end()`

`std::cout << list.get_data()` — like `cout << *itr` (dereference ptr)

`list.next_link()` — like `++itr;`

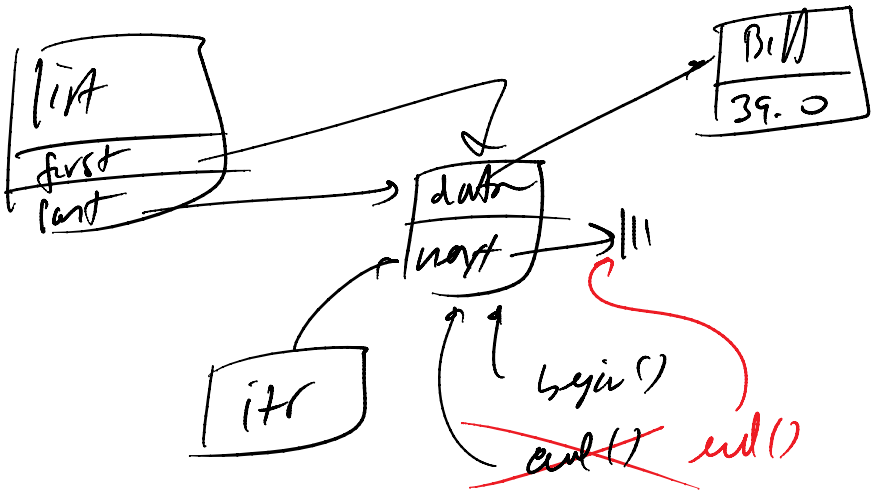
`}`




```
for (itr = list.begin(); itr != list.end(); ++itr)
    std::cout << *itr;
```

expectation: output is "Bill 39.0"

But, begin() == end(), so loop not entered.



would this work?

-Also, want to insert elements onto list:

```
while( !std::cin.eof() ) {
    std::cin >> name >> num;
    if( std::cin.good() )
        list.insert( list.end(), new data_t(name,num) );
}
```

Bill 39.0
McFly 4!3
⋮

idea is to insert at this position (in front of this iterator)