CS2100 : C++ & Lists

End of C++
Simple Lists Intro

Recap:

- HW due today
  - code on hopper or local machine
  - submit via ZyBook
  - git repos – coming soon...
- Lab due Sunday by midnight on ZyBooks
- Reading assignment: due by 2pm on Monday

Last time:

Memory Leaks:

- spaces allocated by program but never deleted.

This isn't an issue with value, pointer, or reference variables.

Problem: new!

The pointer gets deleted, but the data it points at does not.

In a normal program: just rember to delete.

In a class?

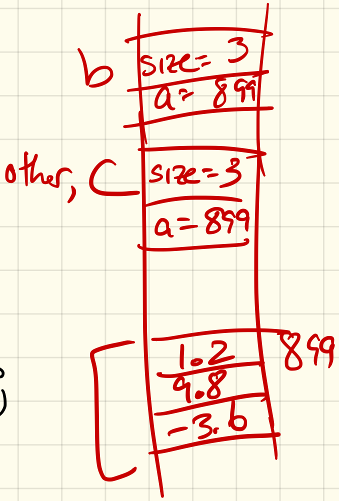
So: Housekeeping functions

Basically, need to deal w/ these pointer issues.

① Copy Constructor

Say I call:

```
MyFloatVec c;
//add data to c
MyFloatVec b(c);
```



Default result?

↳ copies private var:

b's size = c's size

b's a = c's a

Shallow copy

So overriding this:

```
class MyFloatVec {
    //other things...


    public:
        //copy constructor
        MyFloatVec(const MyFloatVec& other)
        {
            size = other.size;
            a = new float[size];
            for(int i=0; i< size; i++)
                a[i] = other.a[i];
        }
```

2 The = operator

Same issue:

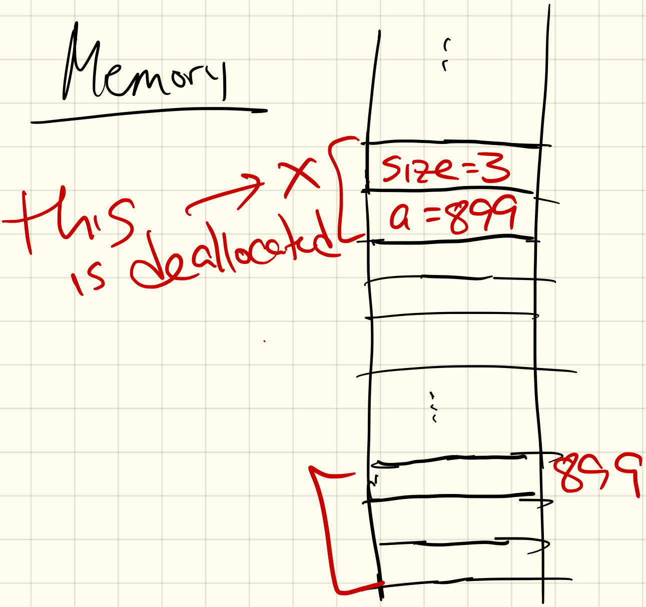
```
MyFloatVec x, b;
//put data in b

x = b;
```

write operator= to fix this
(deep copy)

③ The destructor

Finally: when you create an object

```
int main() {
    myFloatVec x(3);
    ⋮
} // x is destroyed ← what happens?
```

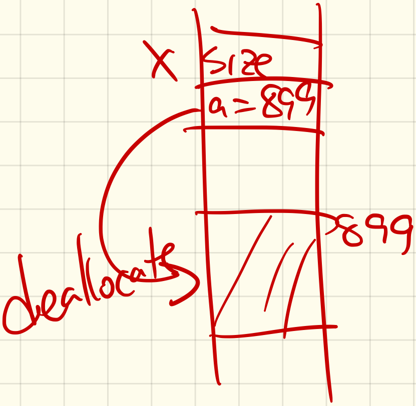


So:

in class:

```
~MyFloatVec() {
    delete[] a;
}
```

opposite of new



Meanwhile:

A few more C++ odds & ends

Enum:

```
enum Color {RED, BLUE, GREEN};

Color sky = BLUE;
Color grass = GREEN;
if (sky == BLUE)
    cout << "It's a nice day!";
```

Reason:

Structs : useful for simple collections of data

```
enum MealPref {NORMAL, VEG, KOSHER};
struct Passenger {
    string name;
    MealPref foodpref;
    bool isFrequentFlyer;
    int freqFlyerNum;
}
int main() {
    Passenger pass;
    pass.name = "Erin Chambers";
    Passenger pass2 = {"John Smith",
            VEG, true, 12345};

  ⋮
 }
```

Templates

If we want a function to work for multiple data types, like ints & floats, use templates.

Ex:

```
template <typename T>
T min (T a, T b) {
    if (a < b)
        return a;
    else
        return b;
}
```

Then:

Templates in classes

These are important in data structures.

Why?

Actually, you'll use these in the stack lab, likely next Thursday.

Error Handling

In C++, we handle errors by throwing exceptions.

(Exceptions are actually their own classes also.)

Recall: What were the ones in Python?

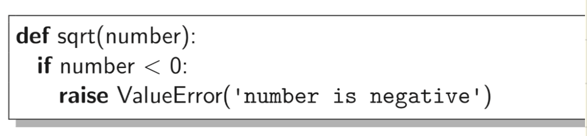
I'll base mine of C++'s default ones:

#include <stdexcept>

↳ see cplusplus for details!

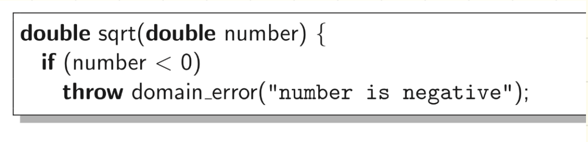
Some examples

In Python:



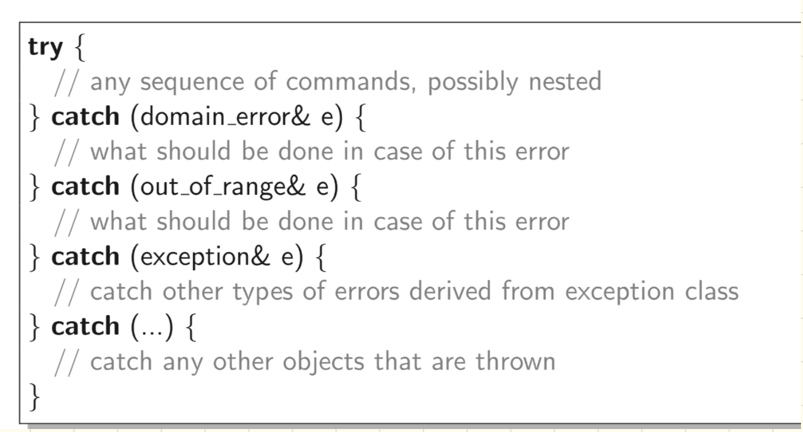
```
def sqrt(number):
  if number < 0:
    raise ValueError('number is negative')
```

In C++:



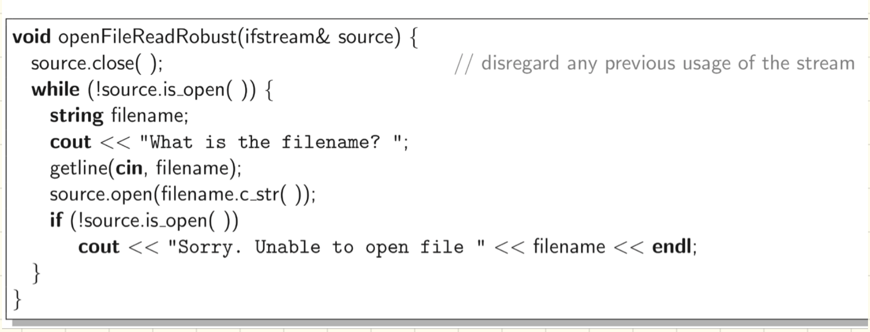
```
double sqrt(double number) {
  if (number < 0)
    throw domain_error("number is negative");
```

In general, to avoid crashing:



```
try {
  // any sequence of commands, possibly nested
} catch (domain_error& e) {
  // what should be done in case of this error
} catch (out_of_range& e) {
  // what should be done in case of this error
} catch (exception& e) {
  // catch other types of errors derived from exception class
} catch (...) {
  // catch any other objects that are thrown
}
```

Reading input example:



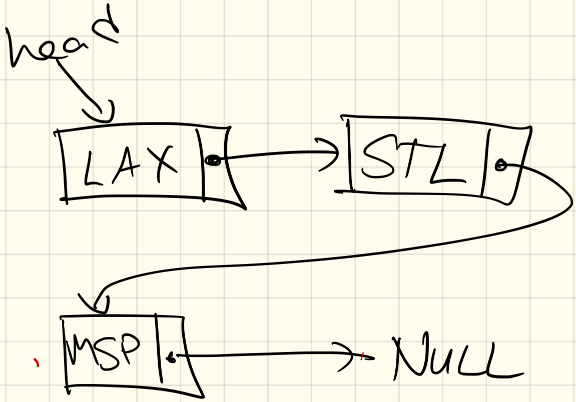
```
void openFileReadRobust(ifstream& source) {
  source.close( );                                  // disregard any previous usage of the stream
  while (!source.is_open( )) {
    string filename;
    cout << "What is the filename? ";
    getline(cin, filename);
    source.open(filename.c_str( ));
    if (!source.is_open( ))
       cout << "Sorry. Unable to open file " << filename << endl;
  }
}
```

Now: A first data structure

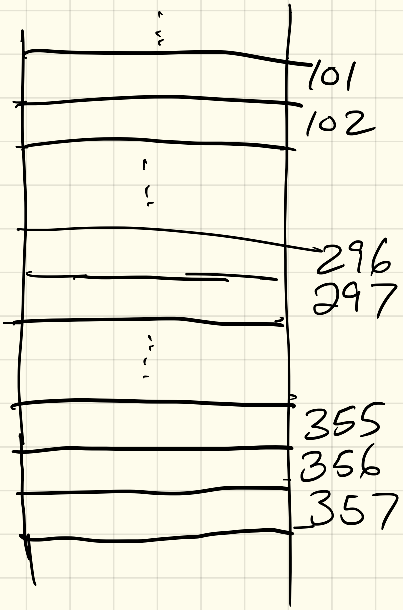
Singly linked lists:

A collection of nodes that have a linear ordering

Ex:



But in memory!



Why this structure?

Note: Not the same as C++'s list class

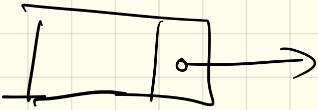
(or Python's, for that matter)

However, this linked structure is useful in a number of data structures.

Why not use an array?

Tradeoff:

Implenention: Nodes



Huh?

We'll need a node struct (or class).

Contents:

Then, in the class, have:

Functions?

Code

```
template <typename Object >
class SLinkedList {
private:
      struct SNode {
            Object data;
            SNode * next;
      }
      int s ;
      SNode* head;
 public:
      SLinkedList();
    ~SLinkedList();



}
```