Slides by Andrew DeOrio

EECS 280

Programming and Introductory Data Structures

Final Exam Review

Schedule

- No Lab next week (last week of class)
- No class on Monday 12/14

Exam time and location

- Time: Thursday 17 December, 10:30am – 12:30pm
    - *Not Michigan time*
- Location – see email or Piazza post

Policies

- Closed book
- Closed notes
- One "cheat sheet"
  - 8.5"x11", double-sided, hand-written, with your name on it
- No calculators or electronics
  - None needed
- Given under the engineering honor code

Study materials

- Practice exams posted on CTools / Google Drive
- Labs
  - Including optional exercises
- Lecture slides
  - Exercises from lecture
- Text book
- Study groups

Topics

- Everything we have covered this semester
- Focus on material beginning with Subtypes and Subclasses

Topics since the midterm

- Subtypes and Subclasses
- Polymorphism
- Container ADTs
- Interfaces and Invariants
- Memory Models
- Copying Arrays
- Deep Copies and Resizing
- Linked Lists
- Templated Containers
- Iterators
- Functors
- Exceptions

Biggest Topics

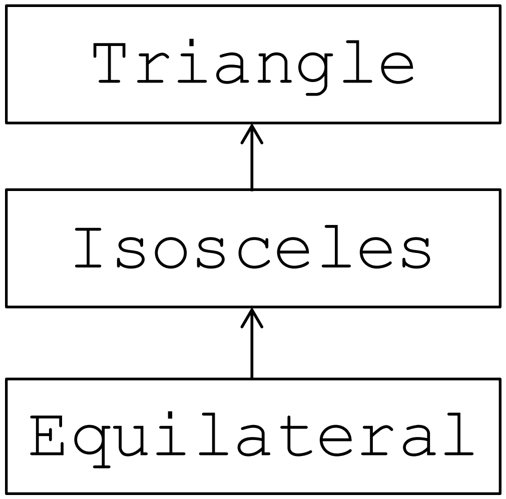
- Subtypes, Subclasses and polymorphism
- Memory Models and deep copies
- Linked Lists
  - Including container ADTs and templates
- Iterators
- Functors
- Exceptions

Subtypes, Subclasses and Polymorphism

"Now Speak!"
"Woof"
"Quack"
"Meow"

stackoverflow.com

Dynamic type



- Other times, the type is not known until run time
- This is called the *dynamic type*

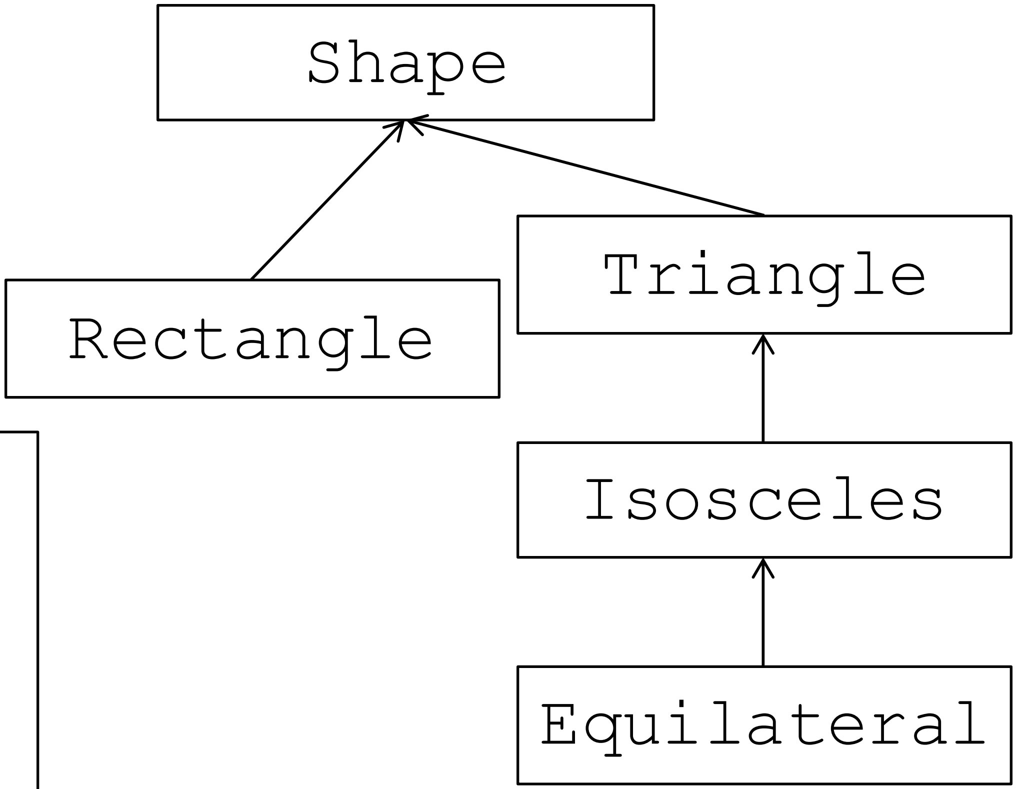
```
//EFFECTS: asks user to select Triangle,
//         Isosceles or Equilateral
//         returns a pointer to correct object
Triangle * ask_user();

int main() {
  Triangle *t = ask_user(); //enters "Isosceles"
  t->print();
}
```

- What is the static type of `t`? What is the dynamic type?

Recall adding the Shape class ...

```
class Shape {
public:
  Shape()
  {cout <<
  "Shape default ctor\n";}
  //...
};
```

```
class Triangle : Shape {
public:
  Triangle()
  {cout <<
   "Triangle default ctor\n";}
  //...
};
```



... etc.

Destructors and polymorphism

```
class Shape {
public:
  Shape() { cout << "Shape default ctor\n"; }
  virtual ~Shape() { cout << "Shape dtor\n"; }
  //...
};
```

- Polymorphic objects need virtual destructors
  - If you have a virtual function and a destructor, then the destructor probably needs to be virtual too.
- Dirtual destructors run in the opposite order as ctors

Exercise: What is the output?

```
int main() {
  Isosceles i;
  Triangle *t_ptr = &i;
}
```

```
int main() {
  Isosceles i(1,12);
  Shape *s_ptr = &i;
}
```

- Assume all ctors, dtors, assignment operators print messages

Exercise: What is the output?

```
int main() {
  Rectangle r;
  Triangle *t_ptr = &r;
}
```

```
int main() {
  Isosceles i;
  Equilateral *e_ptr = &i;
}
```

Memory models and deep copies

Recall our IntSet class

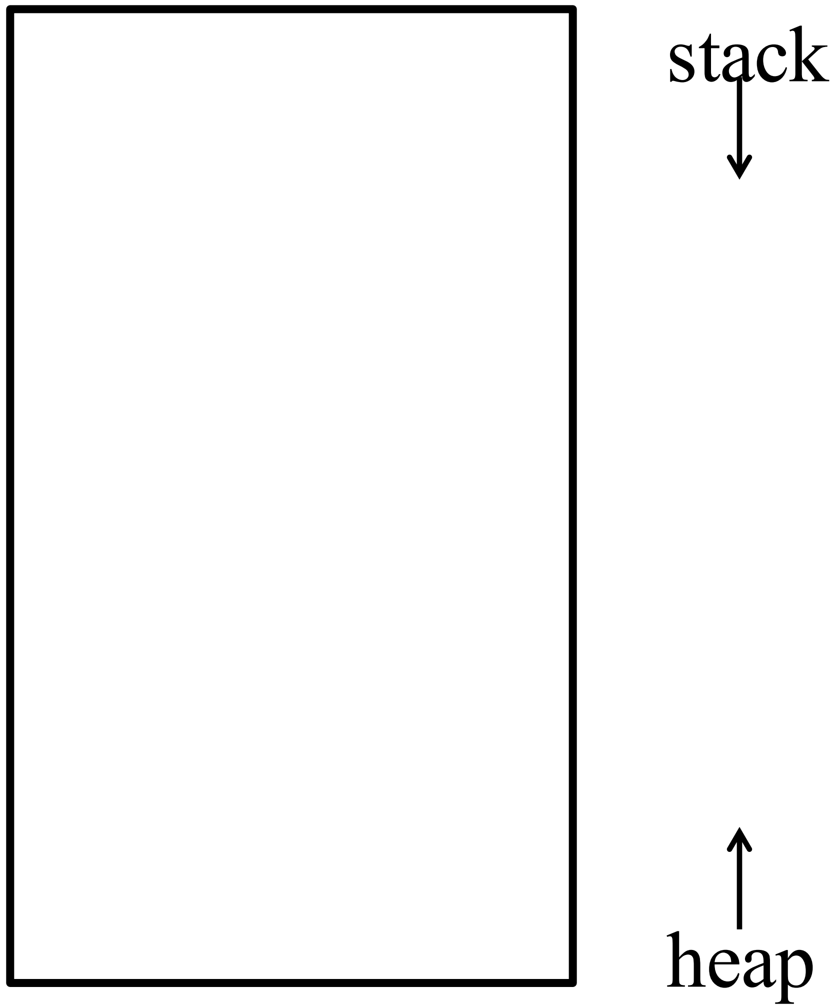
- We used a pointer to a dynamic array to store the set

```
class IntSet {
  int *elts;           //pointer to dynamic array
  int elts_size;       //current occupancy
  int elts_capacity; //capacity of array
  static const int ELTS_CAPACITY_DEFAULT = 100;
  void grow();         // make dynamic array bigger
public:
  ~IntSet(); //dtor
  IntSet(int capacity = ELTS_CAPACITY_DEFAULT);//ctor
  IntSet(const IntSet &other); //copy ctor
  IntSet &operator=(const IntSet &rhs);//assignment
  //...
};
```

Exercise: allocating classes

```
int main() {
  IntSet is1(1);
  is1.insert(42);

  IntSet is2(3);
  is2 = is1;
  is2.insert(43);
}
```

- Draw the stack and heap
- Assume copy ctor, assignment operator and dtor are correctly implemented to do a deep copy



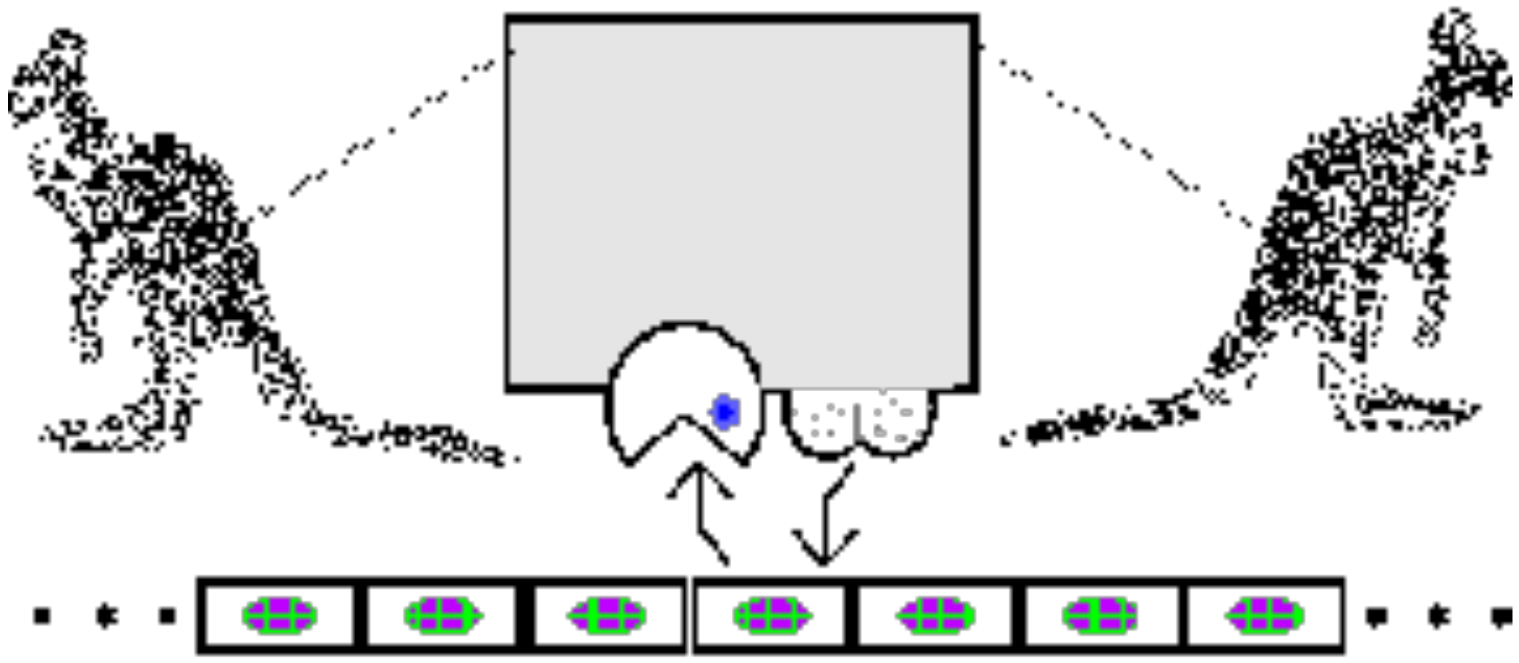
Linked Lists

- You've gotten a lot of practice with these on Project 5, so we will focus on other topics today

prev->next = toDelete->next;
delete toDelete;
// if only forgetting were
// this easy for me.
SNIFF
assert "It's going to be okay.";

http://xkcd.com/379/

Iterators



- I have no idea what this is, but it's in an STL tutorial on Iterators from Brown University

Iterators

```
class Gorilla {
public:
  Gorilla(const string &name_in);
  ~Gorilla();
   string get_name();
};
```

```
int main() {
  List<Gorilla*> zoo;
  zoo.push_front(new Gorilla("Colo"));
  zoo.push_front(new Gorilla("Koko"));

  //add code to print the name of each Gorilla

  //add code to delete each dynamic variable
}
```

Functors

Functors

- Assume you have a functor called `GreaterN` that returns true if an input integer is greater than an input limit
- Assume you have a functor called `LessN` that returns true if an input integer is less than an input limit
- Write a functor called `InRange` that returns true if an input integer is within an inclusive range: `[min, max]`
  - Overload the function call operator
  - Create a constructor to initialize `min` and `max`
- Write a `main()` function that uses instances of `LessN`, `GreaterN` and `InRange` to check if water is a `solid`, `liquid` or `gas` at a given temperature
  - Read an integer `temp` from standard input
  - Freezing point = 32, boiling point = 212

Exceptions

IF THERE IS ALWAYS AN EXCEPTION TO THE RULE
IS THERE AN EXCEPTION TO THAT?
DIYLOL.COM

```
class Error {
  string msg;
public:
  Error(string s) : msg(s) {}
  string get_msg() { return msg;}
};
```

```
void goodbye() {
  cout << "goodbye!\n";
  throw Error("goodbye error");
  cout << "goodbye() returns\n";
}
```

```
void hello() {
  cout << "hello world!\n";
  try { goodbye(); }
  catch (Error e)
  { throw Error("hello error");}
  cout << "hello() returns\n";
}
```

```
int main() {
  try {
    hello();
    cout << "done\n";
  } catch (Error e) {
    cout << e.get_msg()
         << endl;
  } catch (...) {
    cout << "Unknown error"
         << endl;
  }
  cout << "main() returns\n";
  return 0;
}
```

what is the output?

Handling errors

- Write the Triangle constructor so that attempts at creating illegal Triangles are prevented and an error is communicated to the caller. Fix main() as well so that the user is repeatedly asked until a valid triangle is successfully created.

```
Triangle::Triangle(double a_in, double b_in, double c_in);

int main() {
   Triangle *t = 0;
   double a, b, c;
   cout << "Please enter the sides of a triangle" << endl;
   cin >> a >> b >> c;
   t = new Triangle(a, b, c);
   delete t;
   return 0;
}
```