CS180 - Classes in C++

Note Title 9/8/2011

Announcements

- HW1 due Sat.
- Look for HW2 on website soon.
- Lab tomorrow (already posted).

also: global variables

A note on variable scopes:

```
int main () {

    int a;          <- a is created
    cin >> a;
    if (a > 0)
        int b = 12;
    else
        int b = 16;
                    <- b is destroyed
    cout << "a is " << a << endl;
    cout << "b is " << b << endl;   <- undefined variable error

}   // a is destroyed
```

Classes

What is a class? very useful!

- contain functions
- collection of related data

Creating an instance of a class

Example:

```
string s;                  // calling default constructor
string greeting("Hello");  // has input param for constructor
```

Never:

```
string s( );
```

Why? empty function called s

Never:

```
string("Hello") greeting;
```

Why? compile error

Example:

```
class Point {
private:
  double _x;                    // explicit declaration of data members
  double _y;

public:
  Point( ) : _x(0), _y(0) { }   // constructor

  double getX( ) const {        // accessor
    return _x;
  }

  void setX(double val) {       // mutator
    _x = val;
  }

  double getY( ) const {        // accessor
    return _y;
  }

  void setY(double val) {       // mutator
    _y = val;
  };
```

class gets braces

explicitly declare class data

Classes:

① Data – public or private – is explicitly declared, not just used in constructor.

main won't see it.

This is done inside the class, but not inside a function.

Why?

Scope would end in function.

② Constructor Function

- name: always same as class

- no return type

- can initialize variables in a list

```
Point( ) : _x(0), _y(0) { }
```

⟺

```
Point() {
    _x=0;
    _y=0;
}
```

```
Point(double initialX=0.0, double initialY=0.0) : _x(initialX), _y(initialY) { }
```

Other differences

(3) No self! Can just use _x or _y, & it immediately scopes to the class attributes.

(There is a "this", but its usage is a bit more complex.)

(4) Access control - public versus private.

Compiler forces this.

functions and data must be set as public or private

(5) Accessor versus mutator:

changes class data

```
double getX( ) const { return _x; }
void setX(double val) { _x = val; }
```

↳ accessor function

- compiler enforces it

```
const double grav = -9.8;
```

Robust Point class : add functionality

another point as input

```
double distance(Point other) const {
  double dx = _x - other._x;
  double dy = _y - other._y;
  return sqrt(dx * dx + dy * dy);          // sqrt imported from cmath library
}

void normalize( ) {
  double mag = distance( Point( ) );       // measure distance to the origin
  if (mag > 0)
    scale(1/mag);
}

Point operator+(Point other) const {
  return Point(_x + other._x, _y + other._y);
}

Point operator*(double factor) const {
  return Point(_x * factor, _y * factor);
}

double operator*(Point other) const {
  return _x * other._x + _y * other._y;
}
};   // end of Point class (semicolon is required)
```

(3,4) + (2,2)

pt1 + pt2 ;

Important things

1) _x + other._x ← allowed only inside the class

2) using operator+ : can say ~~pt1.operator+(pt2);~~
   Point newpt = pt1 + pt2;

3) two versions of *
   in Python, one function: used isInstance

   pt = pt * 2; ← pt.operator*(2)

   pt = pt1 * pt2;

Additional functions

}; //end of Point class

(Not in the class)

2 * pt

```
// Free-standing operator definitions, outside the formal Point class definition
Point operator*(double factor, Point p) {
   return p * factor;                          // invoke existing form with Point as left operand
}

ostream& operator<<(ostream& out, Point p) {
   out << "<" << p.getX( ) << "," << p.getY( ) << ">";      // display using form <x,y>
   return out;
}
```

<2, 3>

Why?

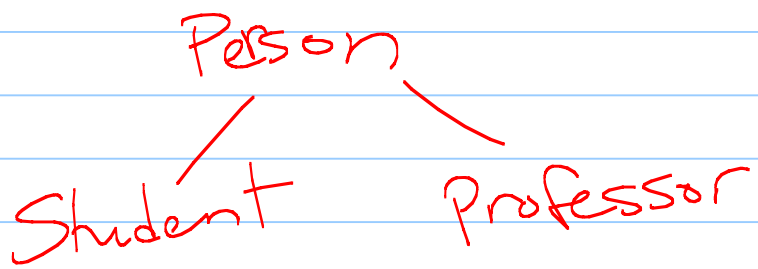
```
cout << pt1 << p2;
cout << "my point is" << pt1;
```

Inheritance

What is inheritance?

A way to be lazy:

child class which steals the data + functions of parent class.



Example: Square class

: parent class

```
class Square : public Rectangle {
public:
  Square(double size=10, Point center=Point( )) :
    Rectangle(size, size, center)        // parent constructor
  { }

  void setHeight(double h) { setSize(h); }
  void setWidth(double w) { setSize(w); }

  void setSize(double size) {
    Rectangle::setWidth(size);       // make sure to invoke PARENT version
    Rectangle::setHeight(size);      // make sure to invoke PARENT version
  }

  double getSize( ) const { return getWidth( ); }
};  // end of Square
```

Rectangle
|
Square

always call parent construction

parent's version

go to parent's version automatically, since nothing of this name here

Other issues

A new type of data. So far, have seen: public and private.

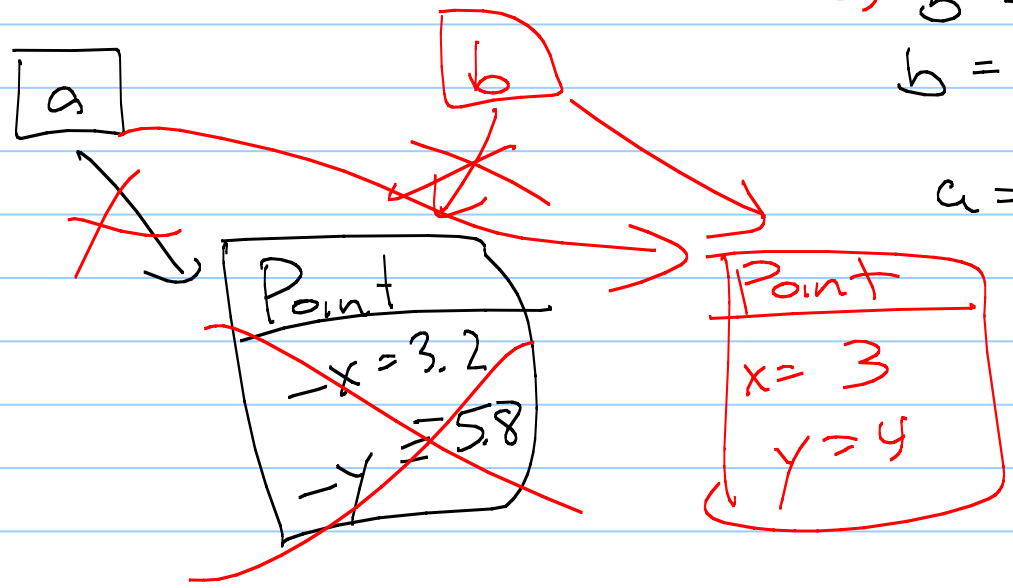
anyone can se — no one can see

What about data that main can't have, but child classes should?

protected:
// variables or functions

Objects

In Python, variables were pointer to actual data.




```
b = a;
b = Point(3, 4);
a = b;
```

C++ : More versatile

C++ allows for 3 different types of variables.

1. Value
2. Reference : &
3. Pointer : *

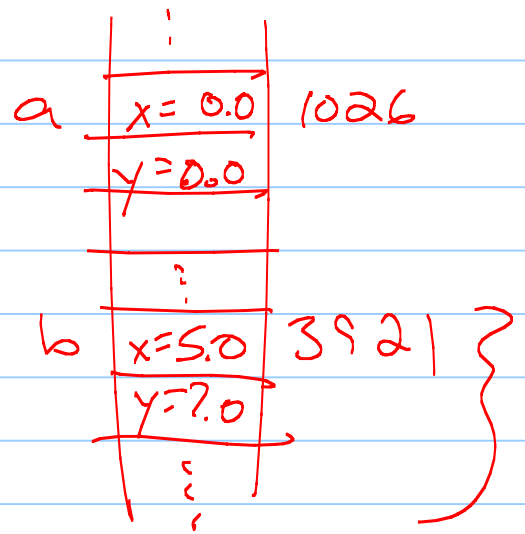
① Value Variables

When a variable is created, a precise amount of memory is set aside.

```
Point a;
Point b(5,7);
```

a : Point
x = 0.0 y = 0.0

b : Point
x = 5.0 y = 7.0



More efficient (for both speed & space).

Now set a = b :

deep copy

They stay separate!

Everything is a deep copy!

Functions: passing by value

```
bool isOrigin(Point pt) {
  return pt.getX( ) == 0 && pt.getY( ) == 0;
}
```

When someone calls isOrigin(myPoint), the value of pt is initialized as a new, separate variable.

Essentially, the line:

Point pt(myPoint);

is run at the beginning of the function!

So do changes to the point last?

No