CS180 — Variable Types

Note Title 9/5/2013

Announcements

- HW due tomorrow
  (email will be hard for me tomorrow!)
- Dept. picnic next week, Wed. at 4pm
- HW2 posted later today
- Tutoring should start next week

Last time

Scoping
Classes:

- syntax
- usage

```
class Name {
 private:
    ...
 public:
    ...
};
```

Inheritance

What is inheritance?

Create "child" class which steals the data + functions from "parent" class.

(a good way to be lazy)

$$\begin{tikzcd}
& \text{FillableShape} \arrow[d, dash] \arrow[dr, dash] & \\
\cdots & \text{Rectangle} \arrow[d, dash] & \text{Circle} \\
& \text{Square} &
\end{tikzcd}$$

int ' value const;

Example: Square 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
```

height width

these are in Rectangle class - overriding those versions

Parent version

Other issues

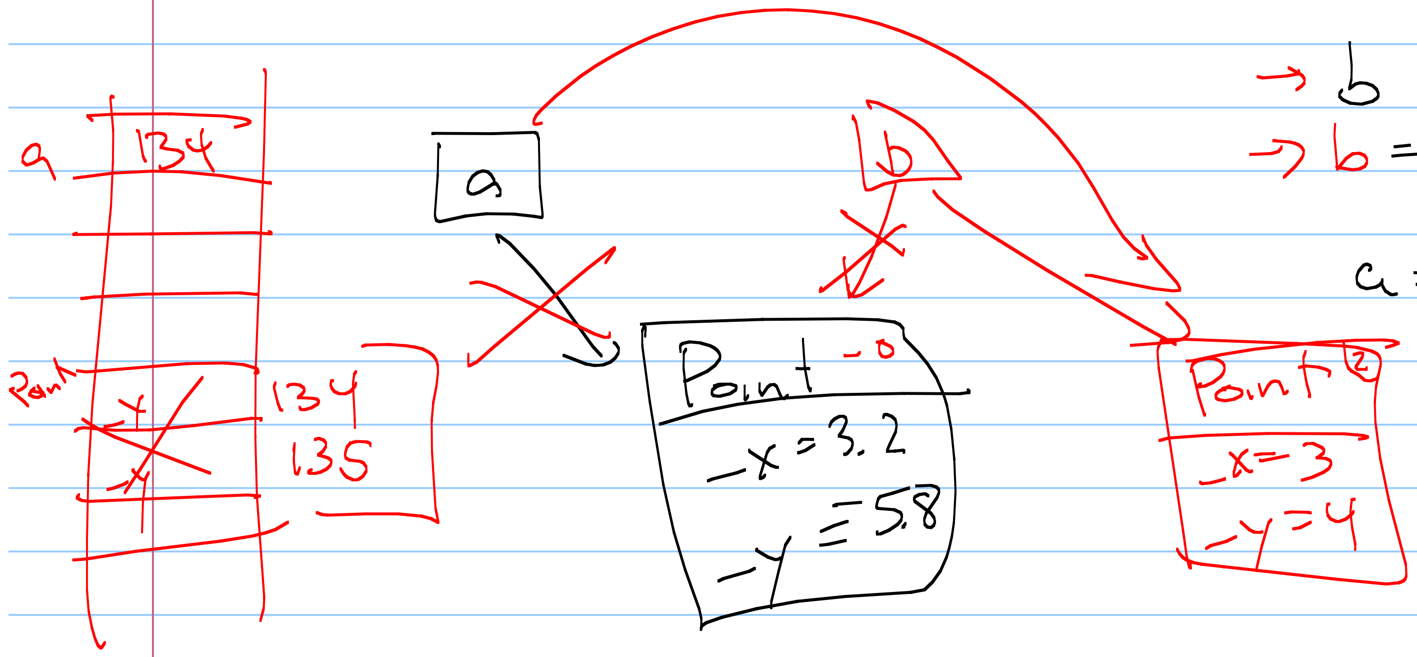
A new type of data. So far, have seen public and private.

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

protected:

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 - what you have seen so far
2. Reference
3. Pointer

① Value Variables

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

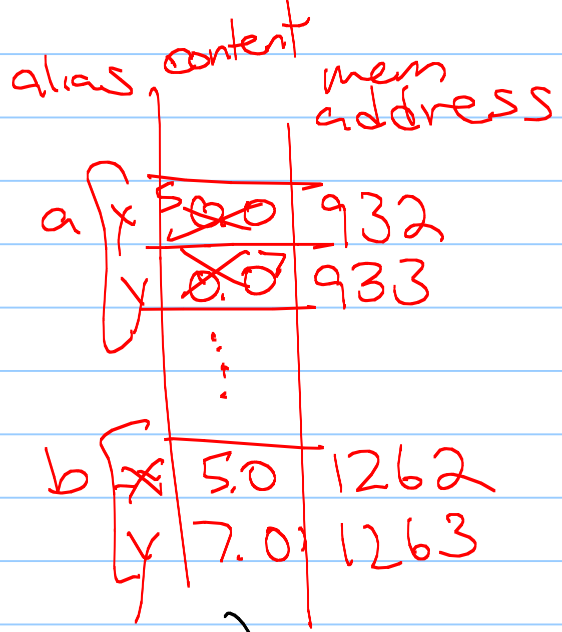
Point a;

→ Point b(5,7);

a = b;

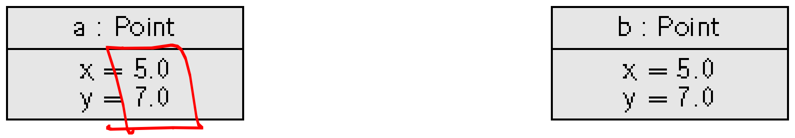
a : Point
x = 0.0
y = 0.0

b : Point
x = 5.0
y = 7.0



More efficient (for both speed & space).

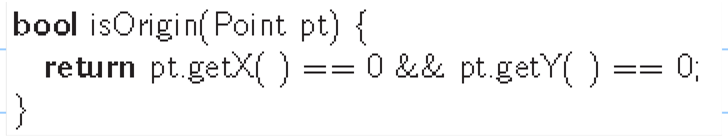
Now set a = b:



They stay separate!

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

② Reference Variables

Syntax: Point & c(a);

- c is created as an alias for a
- More like Python, but c is always 'the' same as a.

Ex: c = b;
Will not make c point to b, but will actually change value of a.

Ex:

```
int  a;        ✓
 a = 35;       ✓
int &  b(a);   ✓
int    c(7);   ✓
b =  63;       ✓
 c = 11;       ✓
a = 50;        ✓
b = c;         ✓
```

name	contents	address
		140
b, a	~~35~~ ~~63~~ ~~50~~ 11	141
c	~~7~~ 11	142
		143
		144
		145
		146
		147
		148
		149
	⋮	⋮

Passing by reference

Reference variables aren't generally use in main.

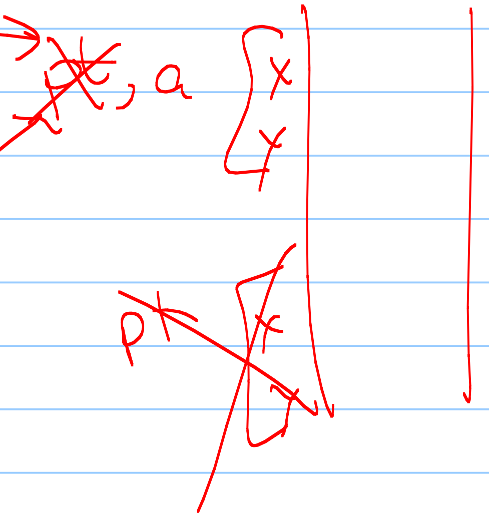
Instead, primary purpose is in functions:

Ex:

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

in main:

isOrigin(a)



Why pass by reference?

3 main reasons

☆ 1) saves time (to copy)

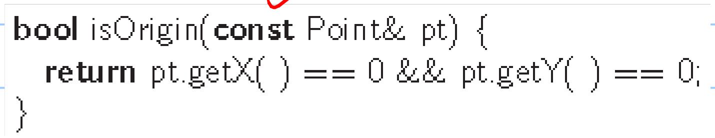
2) saves space

3) Allows changes to persist outside function

"feature"

If we want the speed of passing by reference, but we don't want changes to variable, use const:

const here means pt may not be changed



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

Compiler will enforce that pt isn't changed inside the function.

Ex: setX in function would give an error

Recall: Point output

```
ostream& operator<<(ostream& out, Point p) {
   out << "<" << p.getX( ) << "," << p.getY( ) << ">";
   return out;
}
```

In main:

```
cout << pt <<endl;
```

```
< 5, 7 >
```

Here, & is required since streams cannot be copied.

Note: don't use const. Why?

goal is to change the output stream

③ Pointer variables

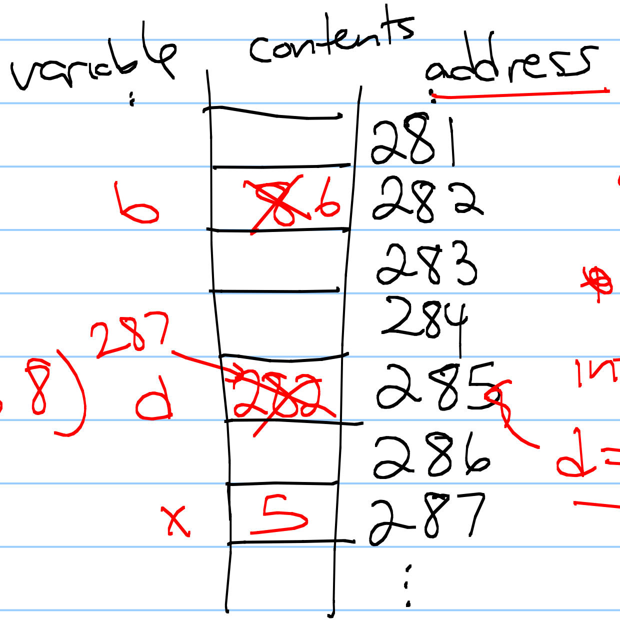
Syntax: `int * d;`

d is created as a variable that stores a memory address.

Ex:

```
int b(8);   ✓
int* d;
         give me address
d = &b;
cout << *d;   (output is 8)
```

But d is not an int.
Can't write ~~d=b~~!



also:

```
*d = 6;
int x = 5;
d=&x;
```