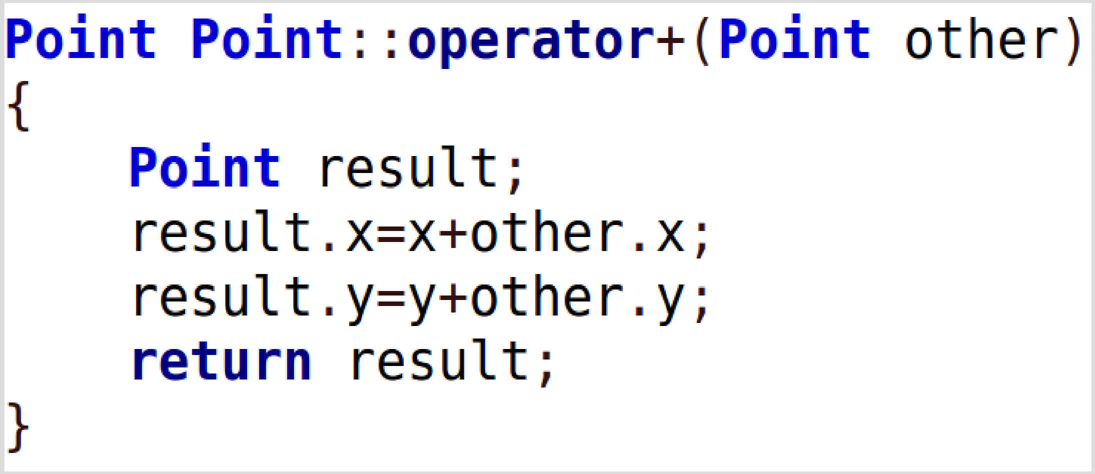
Operator Overload

Ch 11.1

Highlights

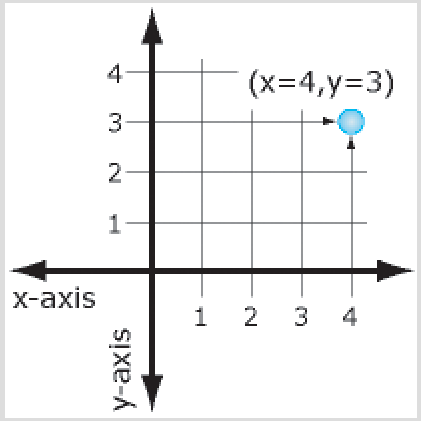
- operator overload

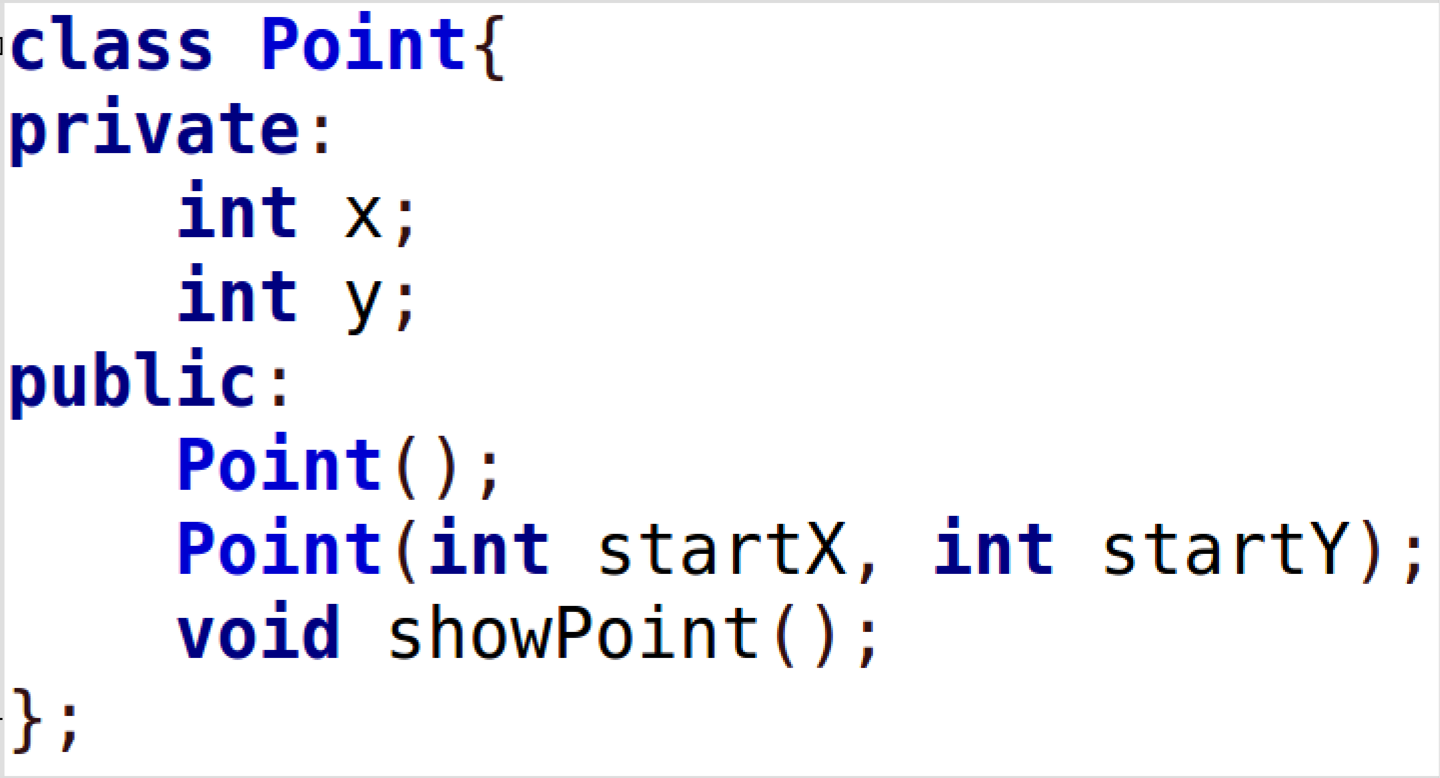


```cpp
Point Point::operator+(Point other)
{
    Point result;
    result.x=x+other.x;
    result.y=y+other.y;
    return result;
}
```

Basic point class

Suppose we wanted to make a simple class to represent an (x,y) coordinate point



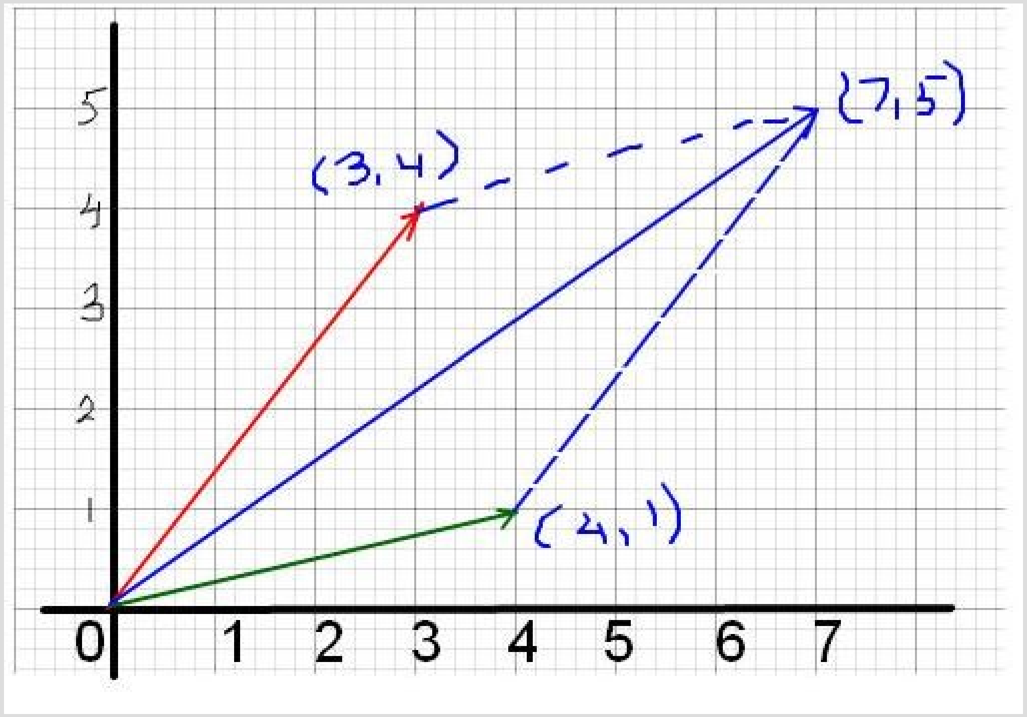


```
class Point{
private:
    int x;
    int y;
public:
    Point();
    Point(int startX, int startY);
    void showPoint();
};
```

(See: pointClass.cpp)

Basic point class

Now let's extend the class and make a function that can add two (x,y) coordinates together (like vectors)



With two ints?

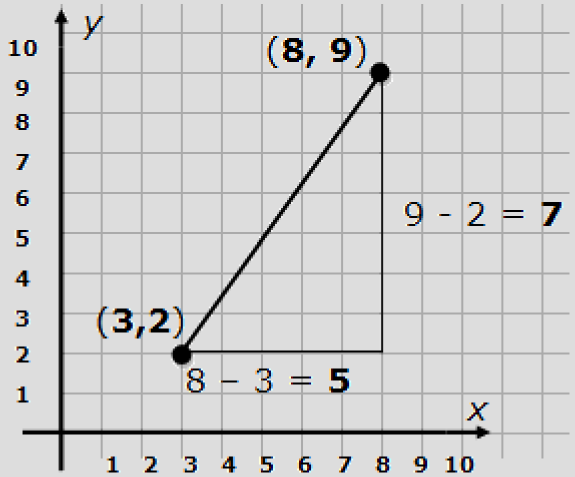
With another point?

(See: pointClassAdd.cpp)

Operator overloading

We can overload the + operator to allow easy addition of points

This is nothing more than a “fancy” function



```
Point Point::operator+(Point other)
{
    Point result;
    result.x=x+other.x;
    result.y=y+other.y;
    return result;
}
```

(See: pointOverload.cpp)

Operator overloading

When overload operators in this fashion, the computer will convert a statement such as:

```
Point c = a+b;
```

function!

... into ...

```
Point c = a.operator+(b);
```

... where the left side of the operator is the "calling" class and the right side is a argument

Operator overloading

You cannot change the number of parts to an operator ('+' only gets 2, '!' only gets 1)

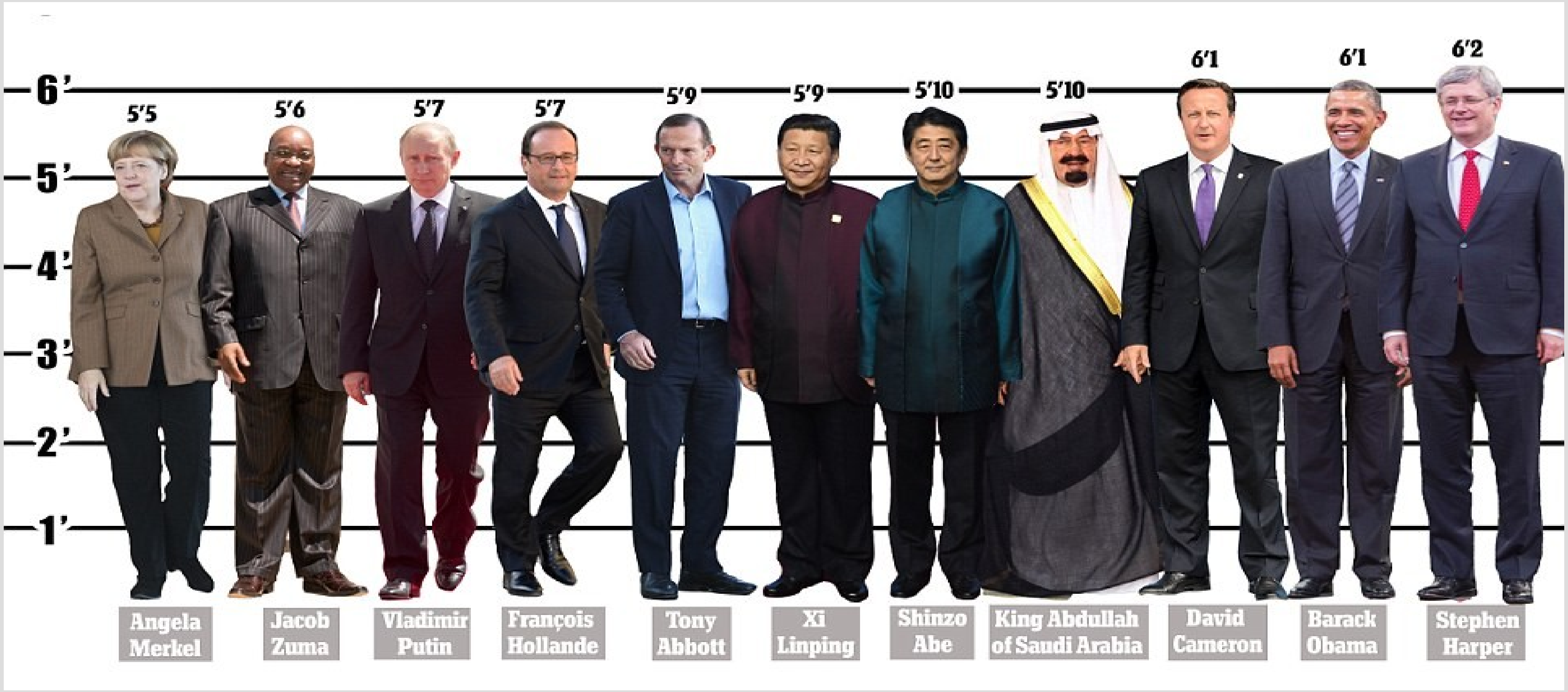
Cannot create “new” operators
(can only overload existing ones)

Cannot change order of precedence
('*' is always before '+')

Operator '=' is special... save for later

Terrible units

Let's make a class that stores people's heights using the terrible imperial units!



(see: heights.cpp)

Terrible units

Write the following operators to compare two different heights:

<

==

>

(see: heightsCompare.cpp)

Operator overloading

Long list of operators you can overload:

( ) // this is normal overloading
+, -, *, /, %
!, <, >, ==, !=, <=, >=, ||, &&
// should be able to do anything above here
<<, >>, [ ]
=, +=, -=, *=, /=, %=, ++ (before/after), --(b/a)
^, &, |, ~, (comma), ->*, ->
^=, &=, |=, <<=, >>=

Operator overloading

Functions define a general procedure (or code block) to run on some inputs

Constructors are nothing but “special” functions that initialize class variables

Operator overloading is a special function that is disguised as a symbol