```cpp
{
    int a = 20;
    int b = 20;

    cout << "a   = " << a << " :   b = " << b << "\n";
    cout << "a++ = " << a++ << " : ++b = " << ++b << "\n";
    cout << "a   = " << a << " :   b = " << b << "\n";
    return 0;
}
```

Here is the output:

```
a     = 20:   b = 20
a++   = 21: ++b = 21
a     = 21:   b = 21
```

Roughly speaking, the notation `a++` means "use the current value of `a` in evaluating an expression, and then increment the value of `a`.

Similarly, the notation `++b` means "first increment the value of `b`, and then use the new value in evaluating the expression." For example, we have the following relationships:

```cpp
int x = 5;
int y = ++x;    // change x, then assign to y
    // y is 6, x is 6
int z = 5;
int y = z++;    // assign to y, then change z
    // y is 5, z is 6
```

The increment and decrement operators are a concise, convenient way to handle the common task of increasing or decreasing values by 1. You can use them with pointers as well as with basic variables. Recall that adding 1 to a pointer increases its value by the number of bytes in the type it points to. The same rule holds for incrementing and decrementing pointers.

**Remember**

Incrementing and decrementing pointers follow pointer arithmetic rules. Thus, if `pt` points to the first member of an