The default copy constructor (provided by the c++ compiler) copies values of member variables to corresponding member variables.

Default copy constructor works fine in most cases.
Problem with member to member copying

• What we get from member to member copying in objects containing dynamic memory (ptrs):

```
IntCell object1(5);
IntCell object2 = object1; // calls copy constructor
//object2.ptrToValue=object1.ptrToValue
object2.write(13);
cout << object1.read() << endl;
cout << object2.read() << endl;
```

What is output?  5  13  or  13

Problem with member to member copying

• Why are they both changed to 13?

• Member-wise copying does a shallow copy. It copies the pointer’s address instead of allocating new memory and copying the value.

• As a result, both objects point to the same location in memory

```
programmer-defined copy constructor
```

Programmer-Defined Copy Constructor

• Prototype and definition of copy constructor:

```
IntCell(IntCell &obj); Add to class declaration
IntCell::IntCell(IntCell &obj) {
    ptrToValue = new int;
    *ptrToValue = obj.read();
}
```

• Copy constructor takes a reference parameter to an object of the class (see text for the reason).

• If the class does dynamic memory allocation, it needs to define its own copy constructor (All linked list implementations)