14.4 Copy Constructors

- Special constructor used when a newly created object is **initialized** using another object of the **same class**.

  - 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

```cpp
Time t1;
Time t2 = t1;
Time t3 = t1;
```

IntCell declaration

- Problem: what if the object contains a pointer?

```cpp
class IntCell
{
    private:
        int *storedValue;  // ptr to int
    public:
        IntCell (int initialValue);
        ~IntCell();
        int read () const;
        void write (int x);
};
```

IntCell Implementation

```cpp
#include “IntCell.h”

IntCell::IntCell (int initialValue) {
    storedValue = new int;
    *storedValue = initialValue;
}

IntCell::~IntCell() {
    delete storedValue;
}

int IntCell::read () const {
    return *storedValue;
}

void IntCell::write (int x) {
    *storedValue = x;
}
```
Problem with member to member copying

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

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

What is output? 5 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

- Prototype and definition of copy constructor:

```cpp
IntCell(const IntCell &obj); // Add to class declaration

IntCell::IntCell(const IntCell &obj) {
    storedValue = new int;
    *storedValue = obj.read();
}
```

- Copy constructor takes a reference parameter to an object of the class
- otherwise, pass-by-value would use the copy constructor to initialize the obj parameter, which would call the copy constructor: this is an infinite loop

Programmer-Defined Copy Constructor

Each object now points to separate dynamic memory:

```cpp
IntCell object1(5);
IntCell object2 = object1; // now calls MY copy constr
object2.write(13);
cout << object1.read() << endl;
cout << object2.read() << endl;
```

Output: 5 13